ABC INFORMATIQUE

N° 22

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

La CAO du micro

Dépister les erreurs

Le disque laser

Le graphisme Atari

EDITIONS ATLAS

M6062-22-12F 85FB-3,80FS-$1.95

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume II, n° 22.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : T.D.F.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juin 1984, 1846. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Point de vue

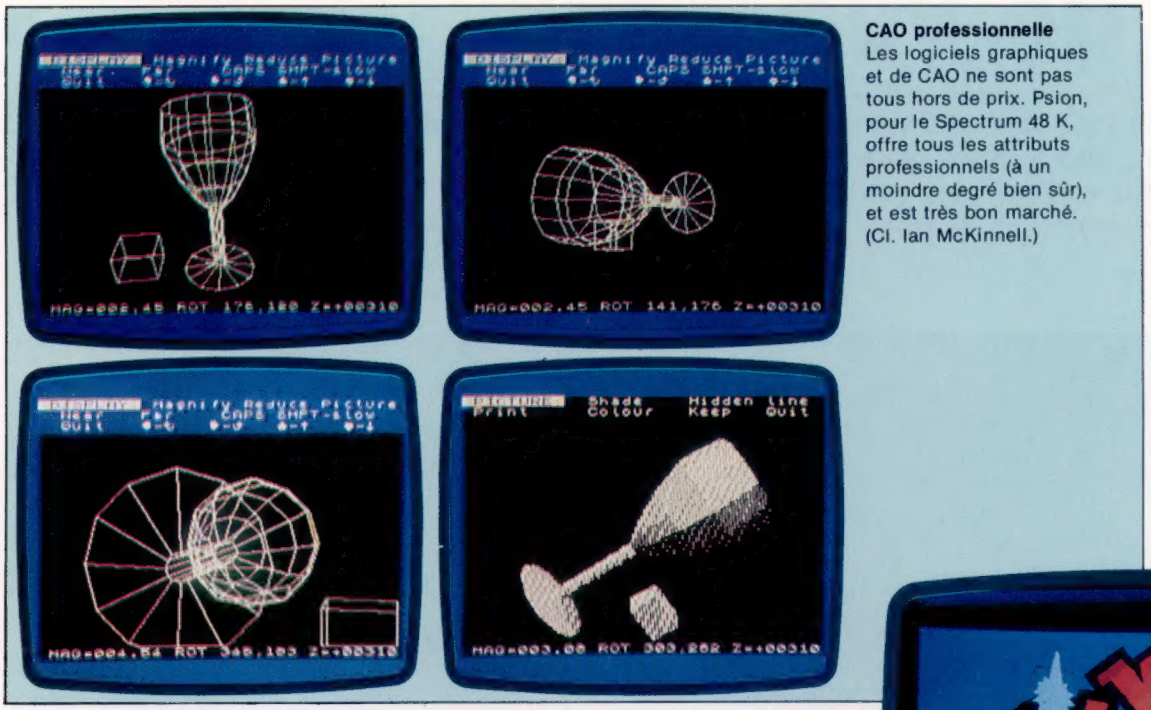
La conception assistée par ordinateur met en œuvre des calculs très compliqués et une qualité graphique de haut niveau. Elle existe pour des ordinateurs personnels.

C'est au Massachusetts Institute of Technology, dans le début des années soixante, que l'idée d'utiliser les ordinateurs dans la conception industrielle est apparue pour la première fois. Mais ce n'est que dix ans plus tard que la technologie informatique put offrir à l'ingénieur la possibilité d'obtenir une représentation de ses créations avec lesquelles il pouvait communiquer (représentation interactive). Un moniteur transmettait instantanément, par l'intermédiaire d'un convertisseur numérique et d'un crayon optique, le résultat du dessin. Ce dernier se faisant, pour ainsi dire, directement à l'écran comme sur un tableau noir. Ces périphériques (le moniteur, le convertisseur numérique et le crayon optique) sont les outils de base de la CAO (conception assistée par ordinateur). La création d'images est semblable à l'animation sur table à chiffrer (voir page 181). L'image est modifiable *via* le crayon optique, ou en incorporant des éléments déjà dessinés et des sous-ensembles déjà conçus. Le dessin fini est transmis à un traceur. L'ordinateur devient un système de dessin comparable, dans ses principes, au traitement de texte, mais où l'image remplace l'écriture.

La création d'images par ordinateur nécessite, comme pour un moniteur, une bonne résolution graphique (qui dépend de la taille d'un élément d'image, ou pixel), ainsi qu'un ordinateur puissant doté d'une grande mémoire pour gérer le moniteur. Ce dernier doit avoir un pouvoir de résolution de 1 000 × 1 200 pixels, et l'ordinateur doit être capable de traiter ce 1,2 million d'éléments visuels en moins de 1/24 de seconde. Au lieu de déplacer des paragraphes, des phrases ou de simples mots d'un texte (insertion, modification ou suppression), comme avec un traitement de texte, un programme de CAO permet de déplacer des éléments d'un dessin dans la page.

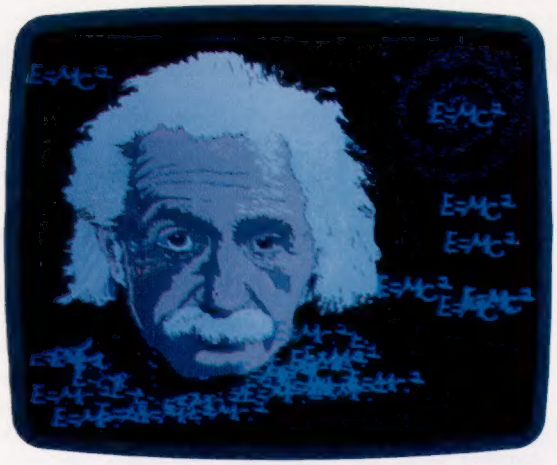
Le principe est donc le même que celui du traitement de texte. L'image peut être travaillée soit en retirant des éléments (sculpture en creux), soit en ajoutant des éléments visuels (sculpture en saillie). Pour réaliser un objet concret, il faut procéder de la même manière : soit en associant des éléments entre eux, comme avec la pâte à modeler, soit comme un sculpteur en retirant des éléments à un bloc de matière. Cette analogie avec un bloc de matière à façonner signifie la présence implicite à l'écran d'un

CAO sur Apple
La créativité de l'utilisateur d'un progiciel de conception assistée par ordinateur a un rôle majeur par rapport aux ressources mêmes du progiciel. Versawriter sur Apple, avec deux lecteurs et un convertisseur numérique, a permis le dessin de cette fleur qui demanda beaucoup de temps d'utilisation machine à un programmeur confirmé. (Cl. Ian McKinnell/ Conny Jude/Soft.)



CAO professionnelle
Les logiciels graphiques et de CAO ne sont pas tous hors de prix. Psion, pour le Spectrum 48 K, offre tous les attributs professionnels (à un moindre degré bien sûr), et est très bon marché. (Cl. Ian McKinnell.)

tableau en trois dimensions. Si le tableau est suffisamment grand pour permettre d'allouer un octet par pixel, chaque élément ou pixel pourra comporter beaucoup d'informations (256 éléments d'information pour un processeur 8 bits, beaucoup plus pour un processeur de 16 ou 32 bits). Mais la création de tant de place mémoire pose un problème pratiquement insoluble. Un compromis acceptable a été trouvé : allouer un bit par élément ou pixel au lieu d'un



CAO haute résolution
Le système Pluto de Io Research permet une image de haute résolution sur de nombreux micros avec, en outre, un processeur rapide et une extension mémoire. Le système de base est relativement peu cher. Il permet huit couleurs fixes et une capacité de résolution de 670 × 576 pixels. (Cl. Io Research.)

octet. La signification de chaque pixel sera alors d'indiquer la présence ou l'absence d'un seul élément d'information visuelle dans l'ensemble figuré.

La conception assistée par ordinateur partage bon nombre d'attributs avec l'image générée par ordinateur : atténuation des courbes, suppression des éléments cachés à l'écran, ombrage, remplissage d'un bloc et recoloriage, par exemple. La solution répétée d'une simple équation, pour une série de valeurs, suscitera la création d'une courbe. Il suffit de spécifier les points de départ et d'arrivée d'une ligne donnée, ainsi que l'éloignement maximal à cette ligne que prendra la courbe, pour fournir une solution à l'équation. Inversement, à partir de cette solution il est possible de déduire l'équation, et de la résoudre pour les valeurs restantes, formant ainsi une courbe.

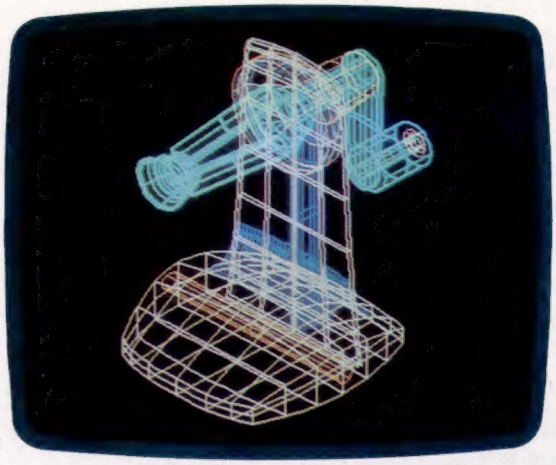
Cette propriété qui consiste à pouvoir composer des dessins à partir de zones constitutives standard définies au préalable représente tout l'intérêt de la CAO.

Par exemple, la conception des circuits imprimés, très complexe, suppose des techniques

d'optimisation afin de mettre en place les composants et leurs réseaux d'interconnexions de la manière la plus économique (sachant que deux réseaux ne peuvent se croiser). La CAO permet donc au fabricant de circuits imprimés de supprimer les tâtonnements en simulant leur réalisation. Le dessin de chaque composant est stocké en tant qu'images prédéfinies et rappelées à volonté. Un dessin de carte pourra être très librement essayé à l'écran jusqu'à ce que la bonne disposition soit trouvée. Il sera ensuite imprimé et servira de base de travail. Cette méthode permet d'assembler des modèles de cartes et d'en comparer l'efficacité, sans être obligé de passer par un support matériel de dessin.

Les circuits intégrés passent par le même système de conception, mais avec une caractéristique supplémentaire : l'agrandissement d'une zone du dessin. En effet, la densité des composants et la complexité des connexions sont telles que les logiciels doivent permettre cette opération. La zone est agrandie, travaillée et replacée dans son contexte d'origine. Cette propriété a considérablement accru l'efficacité de la CAO. Grâce à celle-ci, un seul dessin peut donner une représentation d'un ensemble très complexe, l'échelle étant changée selon le désir de l'utilisateur pour accéder au détail d'une partie selon tous les degrés possibles de variation d'échelle.

Pour un objet encore plus compliqué, une voiture par exemple, ce ne sera plus seulement l'échelle qui variera mais la finesse de la représentation de l'ensemble étudié. Une représentation symbolique des sous-ensembles concernés (systèmes électrique, hydraulique, mécanique, etc.) permettra d'intégrer tous les points de vue particuliers dans une vue d'ensemble. Un code désignera la décomposition de l'ensemble en sous-ensembles (ce pourra être la couleur d'éléments symboliques représentant les sous-ensembles ainsi accessibles).

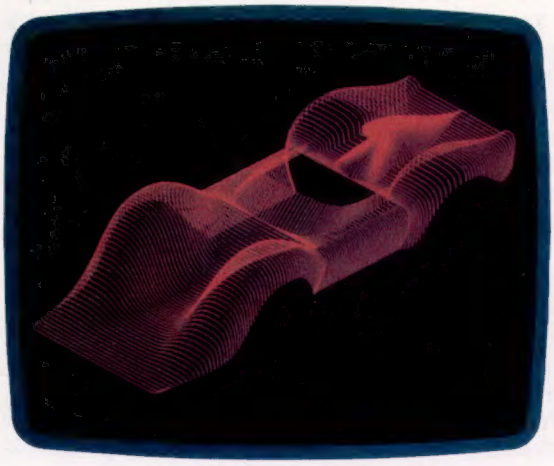


Applicon

Le codage de l'ensemble est une opération qui n'intègre pas seulement la forme ou les apparences. L'abstraction peut être totale : forme et aspect, information sur les matériaux, caractéristiques techniques diverses, prix, etc. Une telle base de données visuelles représente ainsi une percée technologique considérable. Elle peut être consultée pour rechercher une grande variété d'informations, et non plus seulement pour obtenir le tracé des plans du véhicule. Les commandes, le montage, la fabrication des composants ou des pièces, l'intégration linéaire des chaînes de montage, les coûts de revient, le contrôle de la fabrication et beaucoup d'autres aspects deviennent partie d'un système que la CAO gère. Il est tentant d'imaginer l'étape suivante qui sera le suivi complet de la conception assistée par ordinateur, depuis les études jusqu'au contrôle des robots sur les chaînes de production. Il est certain que cette étape est prévisible à très court terme.

Ce type d'application nécessite de gros systèmes informatiques ou, à la rigueur, des mini-ordinateurs très puissants. Mais cela ne veut pas dire que les micro-ordinateurs sont totalement hors course. Ceux qui disposent d'un système d'exploitation CP/M, par exemple, peuvent également faire de la CAO. Les fabricants de micros ont généralement à leur catalogue un progiciel de CAO, même sur des machines peu puissantes comme le Sinclair ZX81. Les résultats que l'on peut en attendre sont modestes puisque la taille et la vitesse d'exécution de l'ordinateur conditionnent la qualité et les performances de l'image créée. Cela dit, les résultats peuvent également être très intéressants et susciter beaucoup d'intérêt.

Imaginez...
L'arrière-plan de cette image, tirée d'une production Lucasfilm « Road to Point Reyes », est dû en grande partie à une nouvelle technique CAO, les fractales. Il s'agit d'accroître la complexité d'une image au fur et à mesure qu'elle approche. Le paysage est d'abord composé de simples formes géométriques décrites en mémoire. Chaque polygone devient de plus en plus complexe en ajoutant sa propre forme sur chacun de ses côtés ; le phénomène se répète de manière aléatoire. Le développement de la forme du flocon ci-dessous illustre ce principe.
(Cl. Siggraph.)



Esquisse de CAO
La première étape de création d'image en trois dimensions consiste en une esquisse. L'image est définie par une suite de coordonnées reliées entre elles par des lignes d'abord droites. Les droites sont manipulées par l'algorithme d'atténuation des courbes ; les lignes cachées sont retirées ; enfin les plans sont coloriés et ombrés afin de créer le relief.
(Cl. Intergraph.)

La machine de Turing

La machine de Turing est un dispositif purement théorique, qui permet de déterminer si un problème est résoluble ou non.

Nous avons jusqu'ici abordé des sujets avant tout pratiques, ou des choses faisables avec votre ordinateur. Mais cette fois nous allons faire un peu de théorie, et aborder ce qu'on appelle l'« ordinatique ». Elle est à l'informatique ce que les mathématiques sont à l'art de l'ingénieur : quelque chose de très abstrait, mais d'où découlent en réalité tous les aspects pratiques.

La machine de Turing, du nom de son inventeur, Alan Turing, est par exemple un concept purement théorique, utilisé dans l'étude des algorithmes. C'est une sorte d'ordinateur réduit à sa plus simple expression; s'il ne peut calculer un problème particulier, on dit que ce problème est non résoluble. Turing estima qu'un tel dispositif devrait comporter trois grandes parties : une mémoire extérieure, qui enregistrerait et stockerait l'information à l'entrée comme à la sortie; un moyen de lire et d'écrire dans cette mémoire; et un organe de contrôle qui déciderait des opérations à suivre.

On admet conventionnellement qu'une machine de Turing dispose d'un ruban (disons magnétique, si cela peut vous aider) de longueur infinie : entendez par là qu'il y en aura toujours assez, quelle que soit la complexité du problème. Ce ruban est divisé en carrés, qui peuvent rester vides ou contenir un symbole quelconque. Une tête de lecture se déplace le long des carrés, dont elle peut lire et modifier le contenu; elle reçoit ses instructions de l'organe de contrôle, qui lui indique ce qu'elle doit y écrire, et dans quel sens se déplacer.

Cet organe de contrôle contient un programme d'exécution; de ce point de vue la machine de Turing a été « construite » spécifiquement en vue d'accomplir une seule application, car elle ne comporte aucun dispositif qui permette de charger ou d'altérer le programme. Nous disons construite entre guillemets, car les seules qui aient été réalisées l'ont été à des fins purement pédagogiques. Il est cependant assez simple d'écrire un programme en BASIC pour micro-ordinateur qui puisse reproduire le fonctionnement de la machine.

L'organe de contrôle est constitué d'un ensemble de « quintuplets », c'est-à-dire d'instructions comportant cinq éléments. L'exécution d'un quintuplet, à n'importe quel moment des opérations, dépend de deux facteurs : le symbole contenu dans le carré lu par la tête de lecture, et l'« état » ou la « condition » de la machine elle-même. Ce dernier point est tout à fait arbitraire : on dira par exemple qu'elle part de l'état S_A et qu'elle s'interrompra lorsqu'elle aura atteint l'état H, les opérations étant alors considérées comme terminées. Entre-temps, l'« état » aura changé de nombreuses fois, en fonction des instructions contenues dans les quintuplets. Cet état reflète ce qui se passe au cours des calculs, et sert à choisir un nouveau quintuplet à exécuter. Disons qu'il fonctionne un peu comme une variable de signalisation en BASIC.

Chaque quintuplet comporte cinq éléments :

1) L'état présent de la machine.

2) Le symbole placé dans chaque carré lu par la tête.

3) Le symbole qui doit être écrit dans ce carré (donc le même que 2 si aucune modification des données n'est nécessaire).

4) Le nouvel état que doit prendre la machine.

5) La direction vers laquelle la tête de lecture doit se déplacer — gauche ou droite.

Le quintuplet (S_A, 5, 3, S_B, D) sera par exemple exécuté chaque fois que, la machine se trouvant à l'état S_A, la tête lit un 5. Celui-ci sera alors remplacé par un 3, la machine passera à l'état S_B, et la tête se déplacera d'un carré vers la droite (D).

Utiliser une machine de Turing pour résoudre un problème quelconque implique d'abord qu'il faut décider de la façon dont les données à traiter seront disposées sur le ruban; même chose pour les résultats, une fois que les opérations auront pris fin (lorsque la machine aura atteint l'état H). Enfin il faut définir l'ensemble de quintuplets nécessaires à l'exécution de l'algorithme.

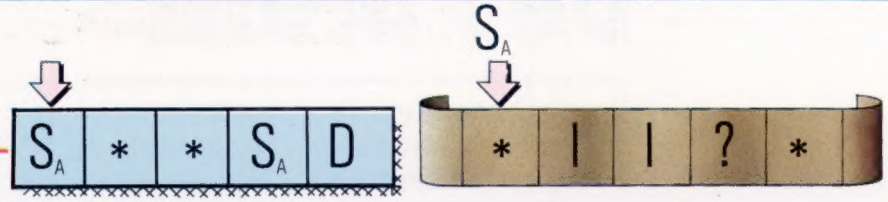
Dans le tableau ci-contre, la machine de Turing accomplit la fonction ET . Nous plaçons deux données de départ (dont chacune est un 1 ou un 0) dans deux carrés adjacents, suivis d'un point d'interrogation qui sera remplacé par la réponse (qui sera également un 1 ou un 0, suivant les données de base). Pour des raisons de clarté nous avons ajouté deux astérisques à chaque extrémité du quintuplet. La machine, à l'état S_A, part de celui de gauche et termine lorsqu'elle arrive à celui de droite.

Il nous faut dix quintuplets en tout, bien que, d'après l'exemple choisi (1 ET 1 = 1), cinq seulement soient utilisés. Si vous essayez d'effectuer l'opération 0 ET 1, par exemple, vous constaterez que cinq quintuplets différents sont sélectionnés.

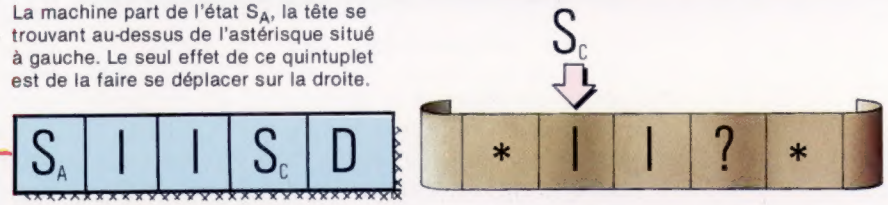
La machine de Turing

Cet exemple montre comment la machine effectue l'opération ET. Les données à traiter sont placées dans deux carrés adjacents, suivies d'un point d'interrogation, qui sera ensuite remplacé par la réponse. Les astérisques placés au début et à la fin de la zone de traitement servent à délimiter celle-ci. Les dix quintuplets ci-dessous sont nécessaires à l'accomplissement de l'opération, bien que, dans chaque cas particulier (ici 1 ET 1), cinq seulement soient effectivement utilisés.
(Cl. Kevin Jones.)

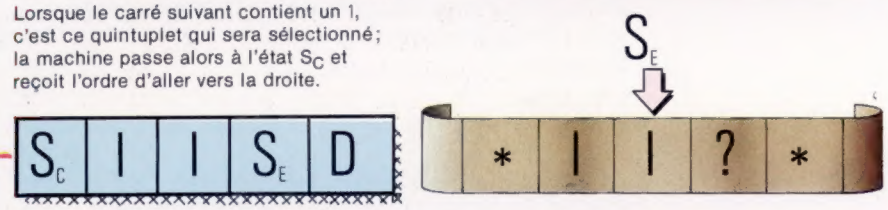
S_A	*	*	S_A	D
S_A	0	0	S_B	D
S_A	1	1	S_C	D
S_A	0	0	S_D	D
S_B	1	1	S_D	D
S_C	0	0	S_D	D
S_C	1	1	S_E	D
S_D	?	0	S_F	D
S_E	?	1	S_F	D
S_F	*	*	H	D



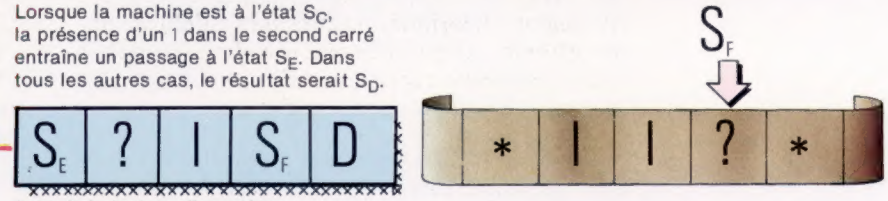
La machine part de l'état S_A, la tête se trouvant au-dessus de l'astérisque situé à gauche. Le seul effet de ce quintuplet est de la faire se déplacer sur la droite.



Lorsque le carré suivant contient un 1, c'est ce quintuplet qui sera sélectionné; la machine passe alors à l'état S_C et reçoit l'ordre d'aller vers la droite.

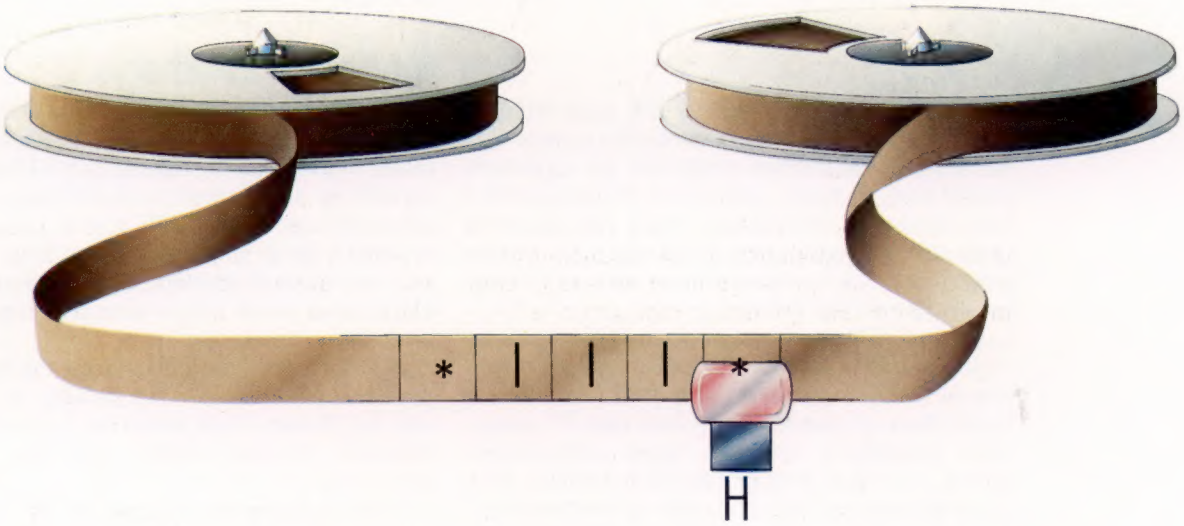


Lorsque la machine est à l'état S_C, la présence d'un 1 dans le second carré entraîne un passage à l'état S_E. Dans tous les autres cas, le résultat serait S_D.



Lors de la lecture d'un point d'interrogation, c'est l'état de la machine (S_E ou S_D) qui détermine le résultat à inscrire à la place, 1 ou 0. Dans les deux cas, la machine passe à l'état S_F.

La machine s'arrête une fois arrivée à l'état H au-dessus du second astérisque. Vous pouvez refaire l'opération pour 1 ET 0, 0 ET 1, et 0 ET 0.



Des bruits

Nous continuons la description de la commande ENVELOPE du micro-ordinateur BBC.

Utilisée conjointement avec la commande SOUND, la commande ENVELOPE est l'une des plus puissantes mises à la disposition du programmeur BASIC. Voyons maintenant la notion de « volume envelope/niveau sonore ». Les paramètres N à NS3, inscrits dans les lignes de programme suivantes, s'appliquent à la hauteur des sons (Pitch Envelope), et ont été traités précédemment.

```
ENVELOPE
N,T,PS1,PS2,PS3,NS1,NS2,NS3,AR,DR,SR,RR,FAL,FDL
```

Les autres paramètres concernent le niveau sonore. Ils servent à indiquer les volumes maximaux et la fréquence de changement de volume sur la durée d'une note, fixée par la commande associée SOUND.

```
AR &DR (-127 à 127) +FAL & FDL (0 à 126)
```

AR donne le niveau d'attaque sonore de la note. Bien qu'une valeur négative soit possible, le choix est compris entre 1 et 127. La fréquence de changement de volume sur la phase décroissante de la note (Decay Rate) est une valeur négative qui provoque une baisse du niveau sonore jusqu'au niveau sonore final (Final Decay Level). Bien que le logiciel permette une gamme allant de 0 à 126 pour ce dernier, les possibilités matérielles courantes n'autorisent qu'une gamme de 1 à 16. Ainsi une valeur de niveau sonore final (FAL) de 50 sera réduite et arrondie à un volume de 6.

```
SR & RR (-127 à 0)
```

Note soutenue (Sustain Rate) et note interrompue (Release Rate) se réfèrent également aux changements de volume par palier de durée, bien que les deux doivent prendre des valeurs négatives. La note est soutenue tout au long de la durée indiquée par la commande SOUND. Ce qui signifie que, si la durée d'attaque et celle de déclin de la note sont ensemble supérieures ou égales à la durée attribuée, il n'y aura pas de phase « note soutenue/sustain phase ».

Et même si cela avait été prévu. La phase « note interrompue/Release » a lieu lorsque la

Des ondes

Le graphisme Atari est devenu un standard que d'autres fabricants suivent.

Les ordinateurs Atari 400 et 800 sont célèbres pour leurs cartouches enfichables; mais les machines elles-mêmes disposent de capacités graphiques en BASIC tout à fait remarquables : neuf niveaux d'affichage, trois modes texte (avec diverses tailles de caractères) et six modes graphiques. La résolution maximale de l'image est de 320 × 192 points.

L'Atari offre un total de seize couleurs, mais cinq seulement peuvent être affichées en même temps. Les jeux de caractères ASCII standard, majuscules et minuscules, ainsi que 37 caractères graphiques spéciaux Atari sont disponibles. Les caractères peuvent être utilisés dans des déclarations PRINT pour des affichages basse résolution et pour des tableaux. Les deux modèles permettent le contrôle du curseur depuis un programme BASIC. Cela se fait en utilisant des caractères de contrôle du curseur dans des déclarations PRINT pour positionner le texte qui suit à l'écran. Ces caractères autorisent les déplacements haut/bas; gauche/droite.

L'une des caractéristiques les plus attrayantes des micros Atari est la définition par l'utilisateur de figures graphiques, connues sous le nom de « missile-joueur ». Celles-ci permettent l'écriture en BASIC de jeux très rapides. Il n'y a pas, néanmoins, de commandes spécifiques pour manipuler ces figures, et tout a lieu en RAM, par les commandes PEEK et POKE, sur les positions mémoire.

Modes d'affichage

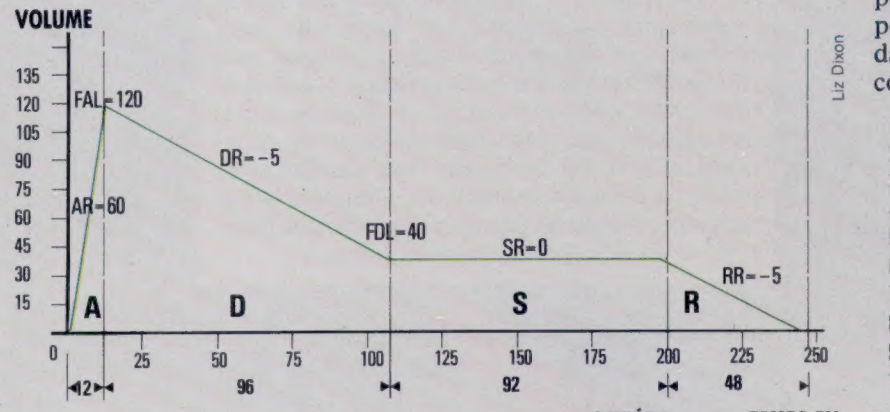
Les modes 0, 1 et 2 servent à l'affichage texte. Lorsqu'on allume la machine, l'affichage est en mode 0 et l'écran est formaté en 24 lignes de 40 caractères. Les caractères d'affichage sont fondés sur le standard ASCII 8 × 8. Les caractères soumis à PRINT en mode 1 sont deux fois plus gros que pour le mode 0, en largeur seulement; alors qu'en mode 2 ils le sont seulement dans la hauteur.

A l'exception du mode 0, les modes graphiques comportent un écran partiel; les lignes du bas de l'écran étant réservées à l'affichage de données diverses telles que les messages d'erreurs.

Pour appliquer la commande PRINT à la partie centrale de l'écran pour les modes 1 et 2, il faut spécifier un numéro d'unité. PRINT # 6 permet

durée de la note est échue. Le volume tombe à zéro selon le rythme voulu, à moins qu'une nouvelle note ne commence sur le même oscillateur. Ce qui signifie que Release est supprimé sauf si « N » a été mis à « 1 » par une autre commande SOUND &.

Niveau sonore



Par rapport au diagramme ci-dessus, les valeurs nécessaires pour créer le timbre ressemblant au piano seraient :

```
T=6   AR=60   SR=0    FAL=120
      DR=-5   RR=-5   FDL=40
SOUND durée=40 (2 s)
```

Ce qui donne :

```
ENVELOPE 1,6,0,0,0,0,0,0,60,-5,0,-5,120,40
```

Le programme suivant utilise tous les sons disponibles avec les commandes BASIC du BBC, pour jouer une séquence bien connue de notes dans l'« enveloppe sonore » du piano, et un court accord final, sur trois notes, répété.

```
10 REM **COSMIC**
20 ENVELOPE 1,6,0,0,0,0,0,0,60,-5,0,-5,120,40
30 ENVELOPE 2,6,1,-1,1,1,2,1,60,-5,0,-5,120,40
40 FOR I=1 TO 4 : READ N
50 SOUND 1,1,N,20:REM**PLAY LA SI SOL SOL**
60 SOUND &1001,0,0,5:NEXT I
70 SOUND &201,2,77,40:REM**ACCORD**
80 SOUND &202,2,89,40:REM**FINAL**
90 SOUND &203,2,109,40:REM**DO MAJEUR**
100 DATA 137,145,129,85:REM**LA SI SOL SOL**
```

l'affichage de texte sur la partie graphique de l'écran. Les modes 3 et 8 sont les modes graphiques et permettent de tracer des points et des lignes à l'écran, selon plusieurs degrés de résolution et avec un certain choix dans les couleurs. Ce tableau indique les options possibles :

Mode	Type	Lignes	Colonnes	Couleurs
0	texte	24	40	2
1	texte	20	20	5
2	texte	10	20	5
3	graphique	20	40	4
4	graphique	40	80	2
5	graphique	40	80	4
6	graphique	80	160	2
7	graphique	80	160	4
8	graphique	160	320	1

Le choix du mode dépendra de la mémoire disponible pour l'affichage. Le mode 5 par exemple demande pratiquement deux fois plus de mémoire pour ses quatre couleurs, que le mode 4 pour deux.

Commandes basic

Les commandes graphiques BASIC sont nombreuses sur les micros Atari 400 et 800. Elles servent également sous une forme modifiée pour les trois modes texte.

```
SETCOLOR a,b,c
```

Quatre registres de couleur sont disponibles, mais inégalement selon les modes. SETCOLOR permet de choisir les couleurs utilisées par ces cinq registres. Dans cette commande, a détermine le numéro du registre de couleurs, 0-4; b est le numéro de la couleur choisie, 0-15; c indique le niveau de luminosité parmi 8 niveaux possibles, en choisissant un nombre pair entre 0-14.

```
COLOR n
```

Cette commande fonctionne différemment selon qu'il s'agit d'un mode texte ou graphique. Pour les modes 0, 1 et 2, n est un nombre compris entre 0 et 255. Dans sa forme binaire, il est constitué de 8 bits : les six premiers se rapportent au code ASCII du caractère soumis à PLOT (tracé); les deux derniers sont réservés à l'information couleur du caractère.

Pour les modes graphiques, n prend une valeur comprise entre 0 et 3, et sert à choisir un registre de commande des couleurs lors du tracé (PLOT) d'un point.

```
PLOT x,y
```

Le point « origine » de l'écran Atari est en haut à gauche. PLOT illumine le point de coordonnées (x, y,). De manière semblable, la commande POSITION :

```
POSITION x,y
```

dispose un curseur invisible au point (x,y) de l'écran.

```
DRAWTO x,y
```

tire une droite (aussi droite que possible pour les modes de résolution les moins performants) entre l'ancienne position du curseur et le point (x,y). Enfin la ligne :

```
X10 18,#6,0,0,«S:»
```

utilise la commande Atari X10 qui permet de remplir ou de peindre une forme à l'écran. Une fois une forme finie, le curseur doit être placé dans le coin inférieur gauche de la forme à colorier. Le coloriage partira du haut pour se terminer au curseur. La couleur est déterminée par POKE 765,C, où C prend la valeur 1, 2, ou 3, selon la valeur prise par COLOR.

Taille de caractères double

Les capacités graphiques des micros Atari sont intéressantes mais assez difficiles à mettre en œuvre. Le choix de couleur est limité, et beaucoup de commandes « standard » haute résolution manquent telle que CIRCLE.
Cela signifie que le programmeur est mis à contribution. En revanche, il existe de nombreux modes texte. Le programme suivant illustre l'utilisation des caractères doubles, avec la commande PRINT.

```
10 REM *LETTRES DOUBLES*
20 GRAPHICS 2+16
30 SETCOLOR 0,3,6
40 FOR X=19 TO 8 STEP -1
50 POSITION X, 1
60 FOR J=1 TO 100: NEXT J
70 PRINT #6;«COURS»
80 NEXT X
90 FOR X=19 TO 6 STEP -1
100 POSITION X,3
110 FOR J=1 TO 100:NEXT J
120 PRINT #6;
  «D'INFORMATIQUE»
130 NEXT X
140 FOR X=13 TO 7 STEP -1
150 POSITION X,9
160 FOR J=1 TO 100:NEXT J
170 PRINT #6;«PERSONNELLE»
180 NEXT X
190 SETCOLOR 0,7,5
200 FOR Y=9 TO 5 STEP -1
210 POSITION 7,Y
220 PRINT #6;«PERSONNELLE»
230 NEXT Y
240 GOTO 240
```

Remarquez que vous pouvez, lors de la sélection d'un mode, supprimer la division de l'écran, en ajoutant 16 au numéro de mode.

Novlangue

Le monde de l'informatique a créé un langage pittoresque. Il est intéressant d'en déterminer l'origine.

Ceux de nos lecteurs qui ont lu des revues informatiques, en français ou en anglais, n'ont pas manqué d'y découvrir certaines expressions bizarres, qui ne sont rien d'autre que le sabir de la profession. Il nous a paru utile de donner l'explication de certaines d'entre elles, dont les origines (anglo-saxonnes), souvent inattendues, ne sont pas sans intérêt.

RONFLANT

BUZZWORD est un terme d'argot pour désigner des termes « ronflants »! A la fin des années soixante, quelqu'un appartenant au service publicitaire de Honeywell mit au point un jeu très simple, le *buzzword generator* (générateur de mots bidons). Il comportait trois colonnes de dix mots chacune, chaque mot étant numéroté de 0 à 9. La première colonne ne comprenait que des noms, et les deux suivantes des adjectifs qui pouvaient se placer en apposition. Il suffisait de choisir trois chiffres et de placer les mots correspondants. On obtenait des expressions telles que « module systémique interactif », de quoi briller dans les salons...

AMORCE

BOOT est une contraction de *bootstrap,* « tirant de botte ». Un *bootstrap loader* (chargeur amorce) est une routine qui s'exécute dès que l'ordinateur est « chargé » (pas question de dire simplement « branché »). Sur les machines dont le système d'exploitation n'est pas installé à demeure dans la ROM, cette routine contient des instructions qui lui permettent d'appeler ce système, logé sur disquette, faute de quoi l'appareil ne pourrait fonctionner.

BIT

BIT, nous disent les dictionnaires, est une contraction de ***BI**nary digi**T*** (chiffre binaire). Mais sans doute vient-il aussi de sa signification anglaise traditionnelle : « Un petit morceau de quelque chose. » Il faut par ailleurs savoir que bit désigne aussi, en argot américain, le huitième d'un dollar, et que les bits vont toujours par deux : deux bits sont ainsi l'équivalent d'un *quarter* (25 cents).

Le mot est souvent employé comme préfixe : *bit-slicing* (découpage en tranches) est un terme qui explique comment certains microprocesseurs évolués peuvent être construits à partir de modules de 2, 4 ou 8 bits, jusqu'à avoir une capacité de 32 bits. Une superstition informatique veut que les programmes qui n'ont pas tourné depuis longtemps présentent de nombreuses bogues (erreurs). D'où le nom de *bit-decay* (*decay :* décomposition).

CLÉ EN MAIN

De nombreuses organisations commerciales ont recours à des firmes spécialisées, chargées d'installer le matériel et de mettre en route les programmes. On parle alors d'une opération « clé en main » **(TURNKEY),** un peu comme dans l'immobilier ou l'automobile : le client n'a plus qu'à tourner la clé et à démarrer. Comme quoi l'arrivée d'un système informatique s'accompagne souvent d'un nouveau jargon.

BASIC

BASIC veut dire officiellement **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (code d'instructions symboliques d'usage général à l'intention des débutants), mais n'est-ce pas une traduction de ce qui est « fondamental »?

MATÉRIEL LOGICIEL

HARDWARE (matériel) et **SOFTWARE** (logiciel) sont typiquement un argot de métier : *hard* (dur) désigne ce qui est tangible, *soft* (doux, mou) ce qui ne l'est pas. On parle aussi de **FIRMWARE** (microprogrammation) pour désigner les logiciels inclus dans le matériel (ROM ou EPROM), et de **LIVEWARE** pour parler du personnel informaticien!

BAUD

BAUD (rythme auquel les données sont transmises) vient d'Émile Baudot, créateur d'un code télégraphique qui sembla un moment l'emporter sur celui de Samuel Morse.

OCTET

BYTE (octet) est un terme informatique d'emploi fréquent; mais, bien qu'il ait à peine trente ans d'âge, on a déjà oublié d'où il vient. Jusqu'à l'apparition du microprocesseur 8 bits, le terme désignait simplement le nombre de bits nécessaires au codage d'un caractère — parfois six, parfois huit. A cette époque les ordinateurs utilisaient rarement des mots de moins de 24 bits, et ceux destinés à la recherche scientifique allaient jusqu'à 64. C'est à partir de *byte,* ironiquement déformé, que s'est créé le terme **NYBBLE** (quartet) qui désigne la moitié d'un octet. **GULP** désigne plusieurs octets.

BOMBE SYSTÈME

Les médias sont toujours à l'affût du jargon à la mode. La délinquance informatique est, de ce point de vue, un terrain favorable : on a ainsi parlé de **LOGIC BOMB** (bombe système) et de **TROJAN HORSE** (cheval de Troie), deux méthodes paraît-il utilisées pour des manœuvres frauduleuses. Une bombe système est un fragment de code appartenant à un programme d'application, et qui reste « endormi » (sans effet) jusqu'à ce qu'un certain délai se soit écoulé, de façon à ce que la fraude soit indiscernable (par exemple lorsqu'il s'agit de faire passer de l'argent d'un compte à un autre). Un cheval de Troie est un programme qui se fait passer pour un autre afin de pouvoir accéder à l'ordinateur.

BOMBE À RETARDEMENT

Une expression analogue, mais qui se réfère à une pratique authentique, est **TIME BOMB** (bombe à retardement). C'est une technique très ingénieuse de protection contre le piratage. C'est un fragment de code inscrit dans le logiciel, qui est désactivé lorsque ce logiciel est acquis par des moyens honnêtes. Sur un exemplaire piraté, cependant, la bombe à retardement attendra le passage d'une certaine date, c'est-à-dire un moment où la compagnie responsable s'est déjà beaucoup servie du programme, qui lui est devenu indispensable. Le lendemain de l'« explosion », tous les fichiers sont inutilisables, et le programme lui-même a été détruit — sauf si la disquette était protégée contre toute réécriture...

FROUILLAGE

GARBAGE (déchets; informations parasites) s'emploie dans plusieurs expressions. C'est ainsi que l'acronyme **GIGO** signifie **G***arbage* **I***n,* **G***arbage* **O***ut,* ce qu'on pourrait traduire par « Comme on fait son lit on se couche » : la qualité des résultats est fonction de la qualité des données entrées.

GARBAGE COLLECTION (rassemblement des déchets; récupération des positions inutilisées) est une opération interne dont votre ordinateur est sans doute capable, pour peu qu'il dispose d'une version du BASIC qui autorise les chaînes dynamiques (qui peuvent changer de longueur durant l'exécution d'un programme). Chaque fois qu'une chaîne augmente de longueur, une nouvelle copie en est faite en mémoire vive. Pour peu qu'on ait un grand nombre d'instructions du type `LET A$ = A$+«*»` (surtout à l'intérieur d'une boucle), la mémoire a vite fait d'arriver à saturation. A ce moment le programme est interrompu momentanément, et une routine inscrite dans la ROM et appelée *garbage collector* (en français programme récupérateur) se met en route, nettoie les zones mémoire concernées, et fait disparaître les fragments de chaînes issus des manipulations précédentes. Le programme principal repart par la suite.

POIGNÉE DE MAIN

Bien des mots du jargon informatique naissent par analogie. Lorsque, en affaires, un contrat vient d'être conclu, on se serre souvent la main. Un **HANDSHAKE** (poignée de main) est le nom donné à un signal électronique qui avertit que la transmission des données est achevée.

0. Réseau	0. Intégré	0. Interactif
1. Potentiel	1. Situationnel	1. Descendant
2. Système	2. Tamponné	2. Linéaire
3. Algorithme	3. Numérisé	3. Analogique
4. Processeur	4. Stochastique	4. Intertâche
5. Tableau	5. Périphérique	5. Compatible
6. Module	6. Heuristique	6. Automatisé
7. Utilitaire	7. Relationnel	7. Significatif
8. Convertisseur	8. Graphique	8. Alphanumérique
9. Générateur	9. Programmable	9. Connectable

Générateur de « syntagmes polysémiques »

Le terme *buzzword* (mot bidon) vient d'un jeu qui permettait de créer tout un vocabulaire technologique très impressionnant, mais parfaitement dépourvu de sens. Vous pouvez, comme nous l'avons fait ici, créer votre propre générateur de formules de ce genre en alignant trois colonnes de dix mots chacune. Un nombre de trois chiffres choisi au hasard vous fournira aussitôt une expression ronflante à souhait.

Le Commodore PET 4032

Premier ordinateur personnel en date, le PET de Commodore a considérablement évolué depuis sa sortie. Ses qualités ne se démentent pas.

Le PET (Personal Electronic Transactor) de Commodore a été à l'origine du boom de la micro-informatique. A sa sortie en 1977, il créa un standard d'une telle perfection que bien des machines plus récentes semblent rétrogrades en comparaison. Le boîtier, tout en métal, est un bon exemple de sa supériorité. En dehors du Memotech et des machines de gestion plus chères, la plupart des micros récents sont moulés dans du plastique, donnant des résultats de qualité douteuse voire carrément mauvaise. L'alimentation incorporée du PET est un autre trait distinctif de la machine.

Des machines de présérie 8 bits et 16 bits ont été disponibles deux ans avant la commercialisation du PET, en kits ou en tant que systèmes minimaux ne comportant que des composants sur une carte imprimée. Le PET fut le premier micro à être prêt à l'utilisation en entrant sur le marché. Il suffisait de le brancher. Les premières versions comportaient un enregistreur de cassettes incorporé avec régulation de la vitesse de défilement, un moniteur incorporé, et un BASIC résident (ROM). Un utilisateur novice n'avait qu'à le brancher pour recevoir ce message rassurant :

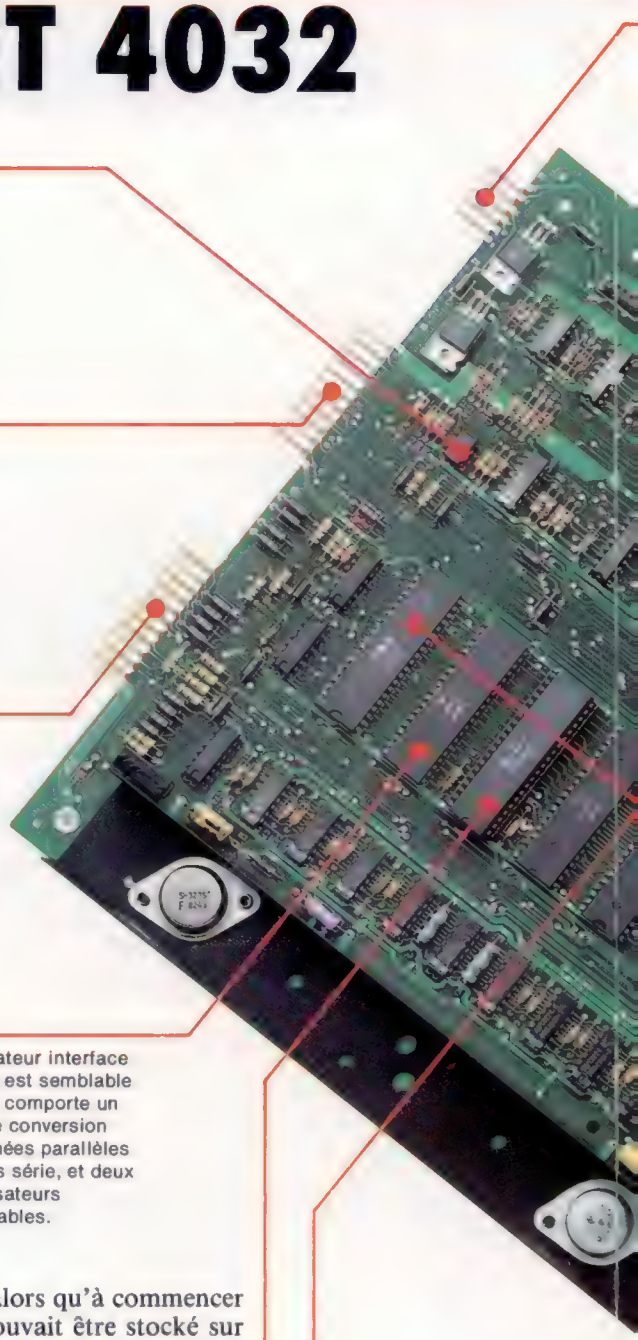
```
COMMODORE BASIC VER. 1.0
7167 BYTES FREE
READY
```

Le clavier et le moniteur du PET
Le premier PET avait un clavier non standard, les suivants avaient l'aspect d'un clavier de machine à écrire avec les touches graphiques sur le devant (sauf pour le modèle de gestion). Tous les modèles ont un moniteur incorporé : les derniers ont un écran de 30 cm avec affichage vert sur fond noir. (Cl. Chris Stevens.)

L'utilisateur n'avait plus alors qu'à commencer de taper, et son travail pouvait être stocké sur cassette, sans être obligé de connecter divers composants ou de charger des programmes systèmes depuis la bande magnétique (ou, pis encore, de les entrer à partir d'un clavier « HEX »... tout un programme!).

Le BASIC Commodore a connu plusieurs révisions depuis ses débuts, et sa dernière version (4.), bien que fondée sur l'originale, a été tellement étendue qu'il s'agit plutôt d'un nouveau langage.

Une autre caractéristique unique du PET est son jeu de caractères. Il comportait à la fois le jeu complet des caractères ASCII, et une grande variété de caractères graphiques. Cela a été mis à profit de manière très créative par les utilisateurs, malgré la faible résolution de ces caractères. Un des problèmes de la machine venait des codes générés au clavier qui ne correspondaient pas au jeu des caractères ASCII, et n'étaient pas ordonnés selon l'ordre standard.



Composant synchronisateur
Lorsqu'on allume l'ordinateur, ce synchronisateur attend une fraction de seconde, après quoi il ré-initialise le microprocesseur sur le début de l'interpréteur BASIC.

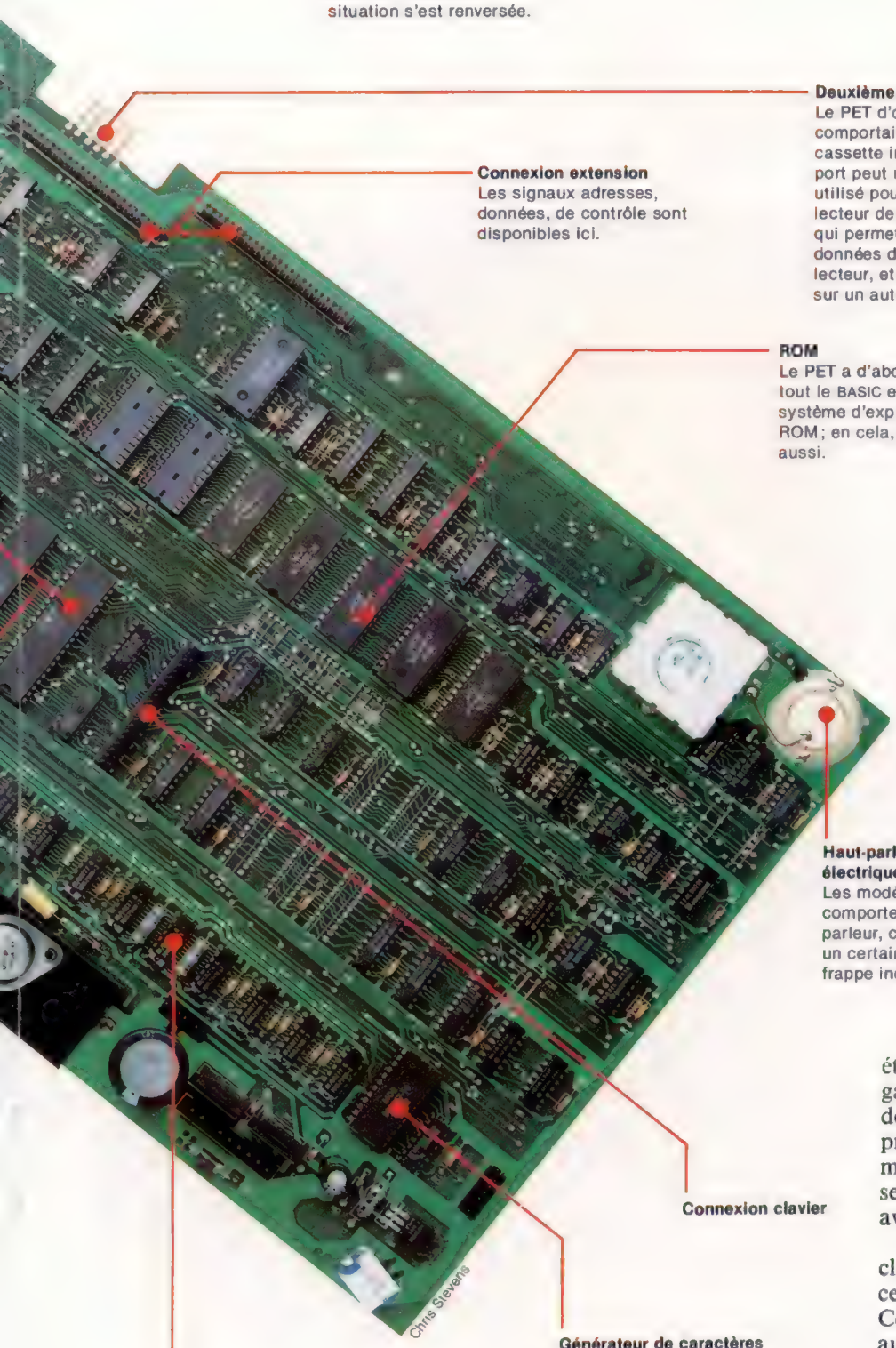
Port utilisateur
Cette interface comporte un certain nombre de lignes très utiles, dont un port parallèle 8 bits, et des branchements pour un moniteur externe. Elle convient tout particulièrement pour une interface domestique électronique.

Port IEEE488
Le PET a été le premier des micro-ordinateurs à comporter cette interface parallèle. Pouvant adresser 15 périphériques, il fut utilisé pour les disques et l'imprimante. C'est également le standard pour les équipements scientifiques de laboratoire.

6522
Cet adaptateur interface polyvalent est semblable au 6520. Il comporte un registre de conversion entre données parallèles et données série, et deux synchronisateurs programmables.

6520
Ces adaptateurs interfaces périphériques se chargent de la plupart des interfaces, dont les cassettes et le clavier.

6502
Le PET a été conçu par Chuck Peddle. Il n'est donc pas étonnant qu'il utilise le microprocesseur 6502 dont Chuck est le père. Bien que les micros de gestion aient des processeurs plus performants, le 6502 reste très répandu.



Port cassette
Le lecteur de cassette est spécial au PET. A ses débuts, il était meilleur que les lecteurs du marché. Depuis, la situation s'est renversée.

Deuxième port cassette
Le PET d'origine comportait un lecteur de cassette incorporé. Ce port peut maintenant être utilisé pour un deuxième lecteur de cassette, ce qui permet de lire les données depuis un lecteur, et de les écrire sur un autre.

Connexion extension
Les signaux adresses, données, de contrôle sont disponibles ici.

ROM
Le PET a d'abord logé tout le BASIC et tout le système d'exploitation en ROM ; en cela, il innova aussi.

Haut-parleur piézo-électrique
Les modèles récents comportent ce haut-parleur, capable d'émettre un certain son lors d'une frappe incorrecte.

Connexion clavier

Générateur de caractères
Outre les 64 caractères alphanumériques, le PET peut générer 64 symboles graphiques. Le texte peut être affiché soit en majuscules, soit en minuscules.

RAM
Le PET comprend en standard de 8 à 32 K. Par une modification spéciale, cette capacité peut être étendue à 96 K.

PET COMMODORE

PRIX
**

TAILLE
480 × 440 × 300.

UNITÉ CENTRALE
6502.

HORLOGE
1 MHz.

MÉMOIRE
32 K RAM.
20 K ROM.

AFFICHAGE VIDÉO
20 lignes de 40 caractères. Écran incorporé de 30 cm vert, moniteur phosphorescent.
256 caractères et symboles graphiques affichables, ou graphisme basse résolution (50 × 80).

INTERFACES
Port utilisateur IEEE488, 8 bits, cassettes (2).

LANGAGE INTÉGRÉ
BASIC, moniteur langage machine.

AUTRES LANGAGES DISPONIBLES
PASCAL, COMAL, LISP.

ACCESSOIRES FOURNIS
Manuel d'utilisation.

CLAVIER
Clavier style machine à écrire : 64 touches avec symboles graphiques sur le devant. Clavier numérique séparé.

DOCUMENTATION
Commodore n'a jamais été loué pour la qualité de ses documentations.

L'utilisation de ces caractères graphiques a été renforcée par la disponibilité de toute une gamme d'imprimantes susceptibles d'effectuer des recopies d'écrans, sans avoir à passer par la programmation en bits de la tête de l'imprimante. Naturellement, cela signifie aussi que seules quelques imprimantes sont compatibles avec le PET, et avec les produits Commodore.

Le résultat de ces originalités est un certain cloisonnement plutôt néfaste au PET, bien que ce dernier ait beaucoup de logiciels en propre. Ceux-ci sont difficilement transposables à une autre machine, et inversement ; mais il y a plus grave : peu de progiciels ont été adaptés pour le PET. Cette machine se trouve donc quelque peu isolée et relativement fermée par rapport à l'évolution du marché. Le PET n'est donc plus en position de pionnier ; mais il garde néanmoins de bonnes implantations. La concurrence, quant à elle, devrait encore s'en inspirer, ne serait-ce que par respect.

Éléments subversifs

Mené avec méthode et de façon progressive, le dépistage des erreurs dans un programme prend moins de temps qu'on le pense.

A mesure que vous devenez de plus en plus « trapu » pour écrire des programmes, vous devez être, par la même occasion, un excellent chasseur de fautes ou d'erreurs. Les erreurs de syntaxe et de logique, que même les programmeurs les plus expérimentés n'arrivent jamais à éliminer totalement, deviennent cependant moins fréquentes et plus faciles à éviter au fur et à mesure que votre expérience s'accroît. Voici quelques suggestions pour vous aider à éviter des erreurs de programmation et vous rendre plus efficace dans le dépistage des erreurs qui se seraient « faufilées ».

Le premier point à contrôler est la conception même du programme. Un programme mal pensé sera certainement rempli d'erreurs. Vous devez d'abord vous représenter le problème de la manière la plus claire possible. Ensuite, vous devez séparer le problème en parties bien distinctes : entrées, sorties, algorithmes, structures de données, traitements... Chaque partie constitue un problème différent. Vous pouvez ensuite subdiviser chaque partie en sous-ensembles. Une approche strictement formelle doit être la règle : l'usage d'un pseudo-langage et d'un organigramme est particulièrement recommandé. Le tout est de pouvoir structurer votre programme de façon à vous y retrouver le moment venu. La frappe au clavier n'interviendra qu'en dernier lieu. Cette méthode arborescente va du général au particulier, elle réduit énormément le temps passé à corriger les erreurs de programmation qui sont inévitables lors de la première rédaction.

La division d'un programme en sous-ensembles vous amène à n'écrire que des sous-programmes et des procédures liées à un programme principal. Vous pouvez ainsi sérier les recherches d'erreurs, et constituer des bibliothèques de sous-programmes sans erreurs. Si vous ne dépistez pas les erreurs de vos sous-programmes une bonne fois pour toutes, vous risquez de les reporter éternellement en réutilisant vos sous-programmes. Ainsi, lorsque vous écrivez un programme servant à trier des données, vous réinventez l'écriture d'une routine de tri, et vous reprenez aussi les mêmes erreurs. Il vaut mieux l'écrire et la vérifier dès le début, puis la garder pour la réutiliser lorsque cela est nécessaire.

Dans les limites où le BASIC le permet, il convient de donner aux variables des noms appropriés, même en abrégé. NET=BRUT-RETENUES, par exemple, est explicite; NT=BR-RT se comprend encore; mais N=B-R est trop ambigu, car on ne sait pas exactement à quoi correspondent ces variables. C'est une bonne méthode que de tenir une table des variables dans la mesure où elle vous rappelle constamment leur existence et leur signification. Cela peut vous permettre de standardiser votre utilisation des variables et d'éviter les mêmes pour plusieurs fonctions.

Cherchez l'erreur

```
100 GOTO 200:X$="VOILA, C'EST FINI"
120 I=12:K=1984
140 FOR K=1 TO LT
150 PRINT"A QUOI BON UNE STRUCTURE ?:N$:NEXT
180 RESTORE
190 FOR L=1 TO I
200 INPUT "DONNEZ VOTRE NOM":N$
220 INPUT"DONNEZ VOTRE AGE":LT
240 GOSUB 100
260 PRINT SI VOUS AVEZ":LT:"MAINTENANT"
280 PRINT"VOUS ETES NE EN":K-LT
300 YR$=K-LT
320 LY=INT(YR)/4*4
340 IF LY=YR THEN IF INT(LY/100)*100=LY THEN GOTO 370
380 PRINT YR CHANGEMENT D'ANNEE" :
    GOTO 420
390 PRINT "YR PAS DE CHANGEMENT D'ANNEE"
400 NEXT
420 PRINT X$
440 STEP
```

Cet énoncé ne sera jamais exécuté, la commande GOTO le sautant.

Ces deux lignes sont dans le mauvais ordre : la ligne 100 devrait comporter : GOTO 190.

K doit contenir une constante, mais cette ligne la supprime.

Les guillemets manquent, NEXT ne sera pas exécuté.

Devrait être : RETURN.

Erreur de syntaxe, le signe ◻ devrait être ◻.

De graves ennuis en perspective... car cela devrait être : GOSUB 140.

Il manque un guillemet.

Le calcul est faux, K ayant changé depuis la ligne 120.

Erreur de syntaxe, devrait être : YR=K-LT.

La dernière parenthèse est mal placée, provoquant une erreur de calcul. Devrait être : INT(YR/4)*4.

Il n'existe pas de ligne 370 !

Devrait être ;» GOTO 420 renvoie à l'extérieur de la boucle FOR NEXT.

Peut avoir besoin du nom de la variable, par exemple, NEXT L.

X$ n'a pas été initialisée, cet énoncé sera sans effet.

Erreur de syntaxe, il devrait s'agir de STOP.

Même avec cette approche méthodique et systématique, il vous faudra passer vos programmes au crible pour en dépister systématiquement les erreurs. Les plus communes sont les erreurs de syntaxe. Vous pouvez toujours les corriger quand vous les rencontrez, mais ce n'est pas aussi facile qu'on le croit :

```
10 PRINT « LES GROSSES ERREURS »
20 PRINT « SE TRAHISSENT D'ELLES-MÊMES »
```

Ces lignes provoquent souvent des messages d'erreur si elles ne sont pas indiquées comme deux lignes distinctes. Si vous oubliez le retour chariot à la fin de la ligne 10, le numéro de ligne 20 suscitera un message d'erreur. Ainsi ce qui était deux lignes dans votre programme sera une seule ligne avec un message d'erreur pour l'ordinateur. Une des manières de dépister ces erreurs est de lister les lignes suspectes séparément, et non comme faisant partie d'un programme.

Les messages d'erreur, quand ils ne sont pas incompréhensibles (comme précédemment), peuvent être trompeurs. Prenons un exemple :

```
25 DATA 10.2,34,56.9,0.008,15.6
30 FOR K=1 TO 5: READ N(K):NEXT K
```

L'exécution échouera à cause d'une erreur de syntaxe signalée pour la ligne 30, alors que l'erreur (une lettre O mise à la place d'un chiffre 0) est en fait à la ligne 25.

Les erreurs de code, non signalées comme erreurs de syntaxe, sont les plus courantes et aussi les plus difficiles à trouver. Il est alors impératif de procéder méthodiquement. Commencez par localiser la partie du programme où l'erreur est susceptible d'être. Cela est relativement facile avec des programmes modulaires et structurés, et peut être tout à fait sûr (ou presque) avec l'utilitaire TRACE. Ce dernier affiche à l'écran la ligne en cours d'exécution. Si votre machine ne comporte pas ce mode, vous pouvez créer des déclarations TRACE régulièrement dans le programme (PRINT « LINE 150 » au début de la ligne 150 par exemple). Vous pouvez de manière semblable utiliser la commande STOP pour interrompre l'exécution sur des passages importants, afin de vérifier les valeurs des variables cruciales. Vous l'obtiendrez soit en mode direct avec la commande PRINT, soit en écrivant un sous-programme s'appliquant jusqu'à la fin du programe :

```
11000 REM AFFICHE LES VARIABLES
11100 PRINT « SCORE,TAILLE,DRAPEAUX »
11200 PRINT SC;SZ;F1;F2
11300 PRINT « TABLE »
11400 FOR K=1 TO 10:PRINT BD$(K):NEXT K
```

Lorsque le programme rencontrera la commande STOP, vous obtiendrez l'état courant des variables en tapant GOTO 11000. Vous pourrez aussi les modifier (en tapant SZ=17, par exemple, et en appuyant sur RETURN), et reprendre l'exécution du programme avec la commande CONTINUE.

Lorsque vous avez localisé l'erreur autour de certaines lignes, ou sur une variable particulière, alors procédez lentement, modification après modification, pour voir l'effet produit sur le déroulement du programme. Il est très facile d'apporter des modifications entre plusieurs exécutions, mais vous ne serez guère avancé en supprimant une erreur quelque part pour en ajouter d'autres ailleurs. En outre, vous aurez vite perdu trace des modifications apportées.

Les boucles et les branchements (particulièrement lorsqu'ils sont imbriqués) sont de véritables nids d'erreurs. Leur écriture comme le dépistage de leurs erreurs devront être l'objet de grands soins.

Considérons le code suivant :

```
460 IF SM<0 AND SC< >-1 THEN IF SC>0 OR
    SM=SC-F9 THEN LT=500
470 FOR C1=1 TO LT:FOR C2=LT TO C1 STEP-1
480 SC=SM+SC*C2
490 NEXT C2:SM=0:NEXT C1
```

Qu'est-ce que cela signifie? Même si vous savez ce que cela doit faire, devinez-vous pour autant si ce code marche ou non? Mettre des déclarations dans une boucle alors qu'elles devraient lui être extérieures est un moyen très sûr de faire des erreurs dans l'écriture du programme.

De même, lors de l'écriture d'expressions conditionnelles IF...THEN, il est absolument impératif de faire figurer toutes les possibilités envisageables. Un cas particulièrement remarquable intervient lorsque vous écrivez des énoncés multiples après IF...THEN. Par exemple :

```
655 IF A$=«» THEN GOTO 980:A$=B$
660 PRINT A$
```

A$=B$ ne sera jamais exécuté. En effet, soit A$«», auquel cas le contrôle passe à la ligne 980; soit A$< >«», et le reste de la ligne 655 est ignoré.

L'expérience est le meilleur atout pour dépister les erreurs dans un programme, mais une démarche progressive et méthodique est d'une valeur inestimable. Prenez votre temps, et pardessus tout, soyez très, très, patient!

Les « bugs »

Une erreur dans un programme est connue sous le nom de « bug » (phalène). La recherche hasardeuse des erreurs et le fait que ce nom renvoie à un insecte en font une démarche vivante. Le programmeur croit avoir affaire à une volonté tenace de dissimulation. Superstitieux, il assimile les erreurs à ce papillon de nuit que le capitaine Grace Hopper a trouvé dans le Harvard II en 1945 en cherchant l'origine d'une erreur.
(Cl. Tony Lodge.)

Le disque laser

Le disque à lecture laser ouvre aux micros deux grandes possibilités : la vidéo interactive et la mémoire de masse.

La plupart des conversations relatives aux micro-ordinateurs tournent autour des dimensions de la mémoire. C'est bien entendu une question importante, mais la capacité de la mémoire de masse (auxiliaire) se révélera plus significative à long terme. Rapidement, un utilisateur sérieux se retrouve avec une pile de cassettes ou plusieurs boîtes pleines de disquettes. Pourtant presque tous ces programmes ne sont jamais modifiés, et il vaudrait mieux les stocker sur des cartouches de ROM que sur un fragile support magnétique. Il faudrait un système à mémoire numérique non modifiable, un peu semblable à une cartouche ROM, mais de plus grande capacité.

Un tel système existe : c'est le disque à lecture laser. Pourtant, il n'est encore utilisé au foyer que comme une alternative au magnétoscope fournissant un matériel préenregistré. Le disque compact est un autre exemple de la même technologie, et commence à remplacer les chaînes hi-fi conventionnelles.

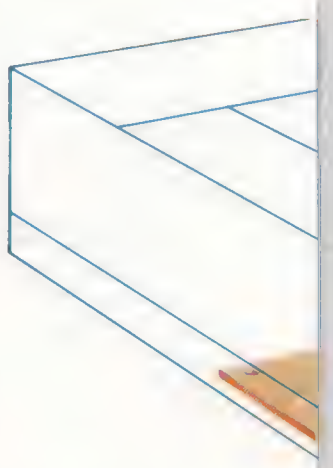
Outre leurs dimensions respectives, ces deux systèmes diffèrent par leur mode d'opération. Le disque vidéo est de type analogique, alors que le disque compact stocke les informations qu'il contient sous forme numérique. Elles sont ensuite délivrées sous forme d'un signal sonore, après passage dans un convertisseur numérique-analogique, qui est d'ailleurs l'inverse du processus à l'issue duquel ces informations ont été collectées. Un disque optique peut d'ailleurs recueillir une quantité d'informations qui peut se chiffrer en millions de méga-octets, ce qui est très supérieur à ce que peut accepter un disque dur de type Winchester.

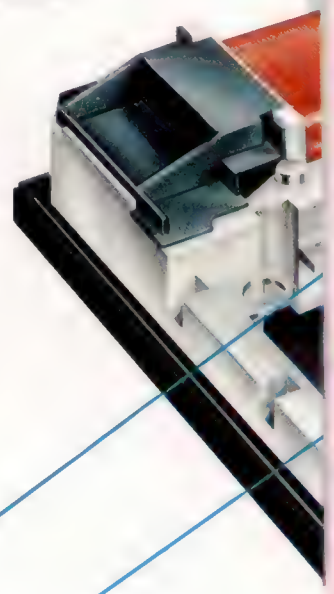
Il existe plusieurs procédés de lecture des disques optiques, mais c'est celui de Philips qui a jusqu'à présent connu le plus grand succès. Il a recours à un disque de 35 cm en plastique, qui n'est d'ailleurs qu'une enveloppe protectrice. Les informations sont en effet stockées au cœur même du plastique, sous forme de multiples dépressions gravées dans une feuille de métal. Comme sur une disquette, elles sont répertoriées et, si l'on dispose du système de lecture approprié, on peut accéder instantanément à telle information particulière : la tête de lecture se place au-dessus d'elle et la lit grâce à un rayon laser. La lumière traverse le plastique et atteint la surface de la feuille de métal; elle est reflétée par les dépressions que contient cette feuille et lue par une cellule photoélectrique. Toute l'information est enregistrée sur un sillon unique en spirale; une image correspond à une rotation du disque. Cela donne un ensemble de 54 000 images par face, soit 36 minutes de spectacle.

Les disques optiques peuvent être utiles à la micro-informatique dans deux domaines. Le premier, qui se développe déjà, est celui de la « vidéo interactive ». Un programme de télévision classique n'est pas interactif : on ne peut pas, par exemple, modifier l'ordre des séquences qui le composent. Ici, toute l'information (textes et images) est stockée sur un disque vidéo, interfacé avec un ordinateur. Le disque peut être employé comme bibliothèque de référence, le texte étant affiché en surimpression sur les images vidéo, le tout sur un écran de téléviseur. L'utilisateur peut, en réponse aux messages émis par l'ordinateur, choisir tel ou tel passage du disque, et le contenu en sera affiché. On peut aussi se servir de l'ensemble pour des essais d'EAO (enseignement assisté par ordinateur) : des images mouvantes ou fixes sont projetées sur l'écran en provenance du disque, l'ordinateur enregistre les réponses de l'élève, en vérifie le bien-fondé et donne une appréciation.

Le problème est en fait celui de l'interface permettant de connecter les deux dispositifs : elle n'est guère disponible commercialement, et de nombreux passionnés ont dû en construire une eux-mêmes. Toutefois Philips met en vente une version professionnelle de son LaserVision, qui permet la vidéo interactive, ou peut être interfacée avec un ordinateur par l'intermédiaire des ports IEEE488 ou RS232.

L'autre domaine prometteur est celui du logiciel. Imaginez qu'on puisse fournir à l'ordinateur, sur un disque unique, quasiment indestructible, un traitement de texte, une base de données, une feuille de calcul et plusieurs douzaines de jeux! Il aurait sans doute le format d'un disque compact; pourtant aucune platine de lecture prévue pour un tel système ne dispose pour l'instant d'une interface. Mais, vu l'énormité du marché potentiel, cela ne devrait pas tarder, et l'on verra sans doute apparaître des modèles spécialisés pour ordinateur. Sony et Philips ont d'ores et déjà annoncé leur intention d'en produire un, qui s'appellera le CDROM.

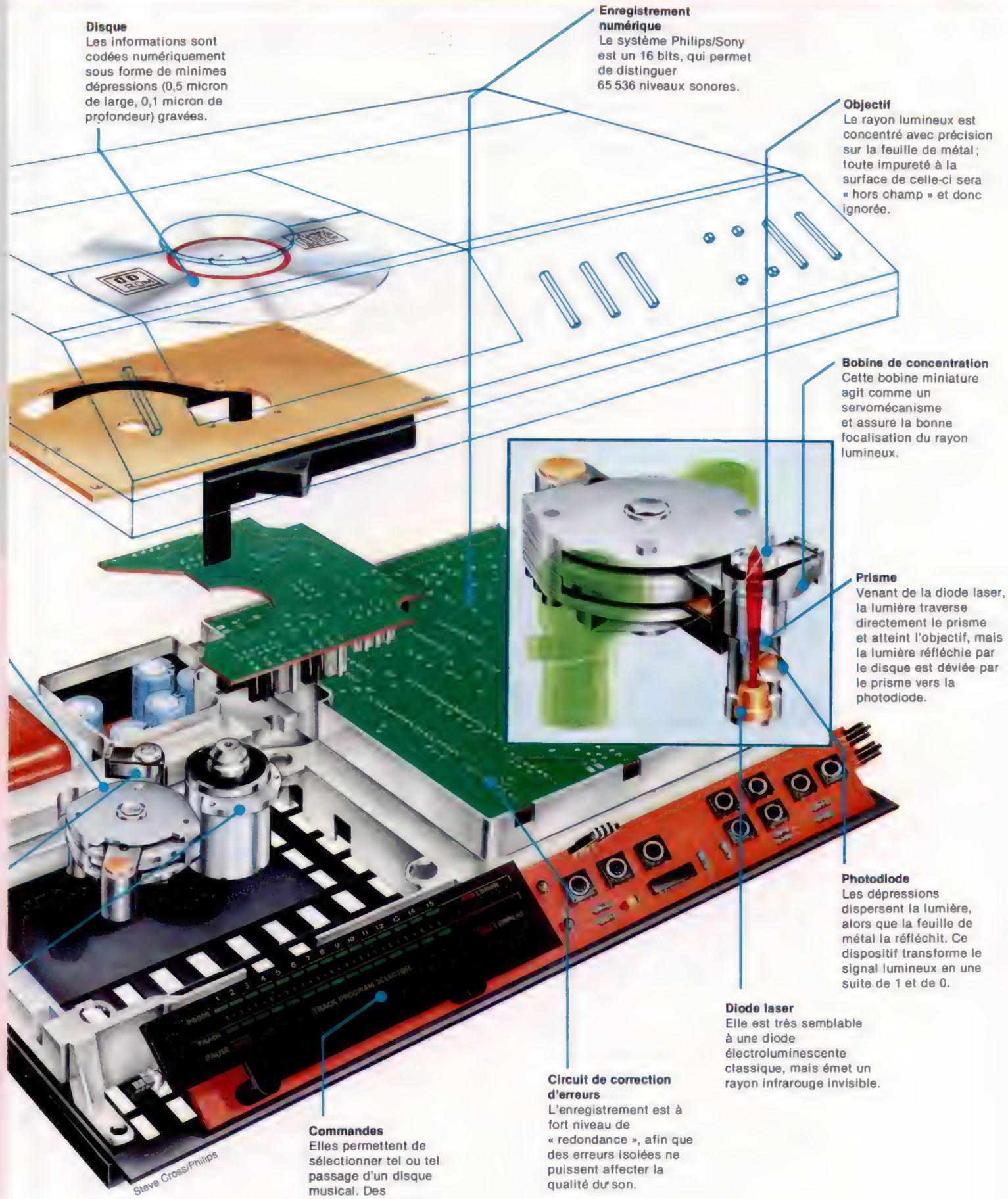




Moteur linéaire
Le servomécanisme qui déplace le bras sur toute la surface du disque est une simple bobine, qui fonctionne en conjonction avec un ressort léger. C'est un dispositif comme on en rencontre sur les appareils de mesure à bobine mobile, comme les voltmètres.

Bras
Le bras pivote librement vers le centre; l'équilibrage est effectué avec beaucoup de soin. La tête de lecture décrit donc un arc au-dessus du disque.

Moteur
La vitesse de rotation du disque est contrôlée de façon très précise. A mesure que le bras passe de l'intérieur à l'extérieur du disque, la vitesse passe de 200 à 500 tours par seconde.



Disque
Les informations sont codées numériquement sous forme de minimes dépressions (0,5 micron de large, 0,1 micron de profondeur) gravées.

Enregistrement numérique
Le système Philips/Sony est un 16 bits, qui permet de distinguer 65 536 niveaux sonores.

Objectif
Le rayon lumineux est concentré avec précision sur la feuille de métal ; toute impureté à la surface de celle-ci sera « hors champ » et donc ignorée.

Bobine de concentration
Cette bobine miniature agit comme un servomécanisme et assure la bonne focalisation du rayon lumineux.

Prisme
Venant de la diode laser, la lumière traverse directement le prisme et atteint l'objectif, mais la lumière réfléchie par le disque est déviée par le prisme vers la photodiode.

Photodiode
Les dépressions dispersent la lumière, alors que la feuille de métal la réfléchit. Ce dispositif transforme le signal lumineux en une suite de 1 et de 0.

Diode laser
Elle est très semblable à une diode électroluminescente classique, mais émet un rayon infrarouge invisible.

Circuit de correction d'erreurs
L'enregistrement est à fort niveau de « redondance », afin que des erreurs isolées ne puissent affecter la qualité du son.

Commandes
Elles permettent de sélectionner tel ou tel passage d'un disque musical. Des périphériques d'ordinateurs utilisant ce disque seront bientôt disponibles.

Steve Cross/Philips

Dernières retouches

En supprimant les anomalies causées par le chaînage des modules et en ajoutant quelques perfectionnements, notre carnet d'adresses informatisé est à présent complet.

Précédemment, nous avions laissé aux lecteurs le soin de comprendre pourquoi, en faisant tourner le programme, en ajoutant (par *AJOUTENR*) et en localisant un enregistrement (par *TROUVENR*), puis en sortant du programme (par *QUITTE*), l'enregistrement ajouté n'était pas sauvegardé. Le problème s'est posé lorsque la variable RMOD a été utilisée pour indiquer qu'un enregistrement avait été modifié (ce qui implique que le fichier pouvait être dérangé). Le sous-programme *TRIENR* devait trier le fichier dans l'ordre alphabétique, puis mettre RMOD à 0 en supposant que le fichier était en ordre. L'exécution de *TRIENR* vérifiait si le fichier était en ordre (RMOD = 0) et ne se souciait pas de le sauvegarder en ordre.

Le fait d'ajouter un enregistrement (par *AJOUTENR*) mettait RMOD à 1 (puisqu'un enregistrement avait été modifié, c'est-à-dire un nouvel enregistrement ajouté), mais *TRIENR* devait mettre RMOD à 0 pour indiquer que le fichier avait été trié. Ce qu'il nous faut en fait, c'est un drapeau indiquant qu'un enregistrement a été modifié et un autre pour montrer si le fichier est trié ou non. Ensuite, des sous-programmes qui ont besoin de savoir que le fichier est trié peuvent vérifier le drapeau « trié », et ceux qui ont besoin de savoir si un enregistrement a été modifié peuvent vérifier le drapeau « modifié ».

Nous désignerons ces deux drapeaux respectivement par RMOD, pour montrer si un enregistrement a été modifié, et TRIE, pour montrer si le fichier a été trié.

Dans un programme précédent, la ligne 1230 contenait l'instruction LET SVED = 0. La variable SVED n'a pas été employée jusqu'à présent, mais, en écrivant cette ligne, nous nous sommes rendu compte que RMOD seul ne suffisait pas. Le nom de variable SVED fut choisi avec l'idée que certaines conditions devraient être vérifiées avant qu'une sauvegarde (sur bande ou disque) ne fût nécessaire.

TRIE serait un nom plus approprié pour ce drapeau (pour indiquer que le fichier est trié). Nous modifierons donc la ligne 1230 comme suit :

```
1230 LET TRIE = 1
```

Il existe donc quatre états possibles quant à la condition du fichier de données :

RMOD	TRIE	
0	0	Non modifié, non trié (illégal)
1	0	Modifié, non trié
0	1	Non modifié, trié
1	1	Modifié, trié

RMOD = 0 et TRIE = 0 est illégal, parce que le programme s'assure que le fichier de données est toujours trié avant de le sauvegarder. Quand le programme tourne, RMOD est mis à 0 (ligne 1220) pour indiquer qu'aucune modification n'a eu lieu, et TRIE à 1 (ligne 1230) pour indiquer que le fichier est trié.

Toute opération qui modifie un enregistrement (comme *AJOUTENR*, *EFFENR* ou *MODENR*) met RMOD à 1, et ce drapeau n'est ré-initialisé par aucune des opérations suivantes. TRIE, dont la valeur initiale est 1, est remis à 0 par toute activité qui pourrait déranger l'ordre des données (comme dans *MODENR* si la zone du nom est modifiée). Toute activité qui a besoin de savoir que le fichier est trié (comme *TROUVENR*) vérifie toujours TRIE et appelle le sous-programme de tri si TRIE=0. Grâce à ces deux drapeaux, au lieu de RMOD seulement, le programme peut s'achever sans sauvegarder le fichier de données si aucune modification n'a eu lieu au cours du programme. On ne risquera pas de terminer sans sauvegarder si un tri a lieu après la modification d'enregistrement.

L'autre variable non utilisée jusqu'ici est ACT. Cette variable sert à sauvegarder la position « actuelle » dans le tableau d'un enregistrement, après qu'on en a localisé un par la routine de recherche. ACT n'est pas remis à 0 après qu'une valeur lui a été attribuée; on l'utilise pour communiquer l'information concernant l'enregistrement cible à d'autres routines du programme. La fin de *TROUVENR* a été modifiée aux lignes 3320 et 3330 pour mettre la valeur de ACT : à 0 si la recherche n'a pas abouti; à MED si la recherche a abouti.

La ligne 13340 se branche sur le sous-programme *PASENR* si ACT égale 0. Celui-ci affiche un message qui dit que l'enregistrement n'a pas été trouvé et donne la clé de recherche, NOMCHP$(TAILLE). *PASENR* retourne au menu principal après que l'on a appuyé sur la touche d'espacement. On pourrait aisément modifier *PASENR* pour permettre à l'utilisateur d' :

```
APPUYER SUR RETURN POUR ESSAYER A NOUVEAU
OU SUR LA BARRE D'ESPACEMENT POUR CONTINUER
```

Il semblerait que le moyen le plus facile pour réaliser cela serait d'appeler à nouveau *TROUVENR* si l'on avait appuyé sur RETURN. Cependant, bien qu'il ne soit pas illicite d'appeler un sous-programme à partir de lui-même en BASIC, cela rend confuse l'adresse de retour et peut faire que le sous-programme se répète même si vous ne le voulez pas. Il y a moyen de

contourner ce problème, mais la programmation commence à être un peu délicate!

Il serait plus simple d'utiliser un drapeau (tel que PENR pour « pas d'enregistrement ») et de le ré-initialiser dans *PASENR*, ce qui permet au sous-programme de retourner normalement et oblige à revenir à *EXECUT* dans le programme principal; par exemple : 95 IF PENR=0 THEN 80. Cette approche a été essayée et elle a marché. Mais le codage commençait à être un peu confus. Conformément à notre principe d'éviter les GOTO, nous avons décidé de rester simples et de retourner directement au menu principal si un enregistrement n'est pas trouvé par *TROUVENR*.

Signalons un petit complément à la ligne 10490 dans *MODNOM*. La variable numérique S devrait aussi être ré-initialisée (LET S=0). Sinon, dans certains cas, *MODNOM* pourrait ne pas fonctionner correctement.

L'autre routine mise en œuvre dans cette version finale du programme est *MODENR*. Cette routine localise d'abord l'enregistrement à modifier en appelant *TROUVENR* (ligne 14120). Cette ligne appelle la ligne 13030, et non 13000, afin de supprimer l'instruction d'effacement de l'écran de *TROUVENR*. Si l'enregistrement ne peut être localisé, le programme retournera au menu principal (à la ligne 14130). S'il est localisé, l'enregistrement cible est affiché à l'écran, ainsi que le message suivant :

```
MODIFIER LE NOM?
APPUYER SUR RETURN POUR ENTRER
UN NOUVEAU NOM OU SUR LA BARRE D'ESPACEMENT
POUR ZONE SUIVANTE
```

La routine qui détermine l'option nécessaire se trouve aux lignes 14190 à 14280.

Les lignes 14190 à 14220 constituent une simple boucle qui ne se termine que si l'on appuie soit sur la barre d'espacement, soit sur RETURN. Si A$ n'est pas CHR$(13) (la valeur ASCII pour le retour du chariot) et pas un espace (ou bien CHR$(32) au lieu de « »), I sera ré-initialisé et la boucle se répétera. Si la touche appuyée était RETURN (c'est-à-dire si la zone du nom doit être changée), les quelques lignes suivantes rempliront NOMCHP$(ACT) avec le nouveau nom, mettront une valeur à RMOD, ré-initialiseront TRIE, appelleront *MODNOM* et rempliront MODCHP$(ACT) avec le nom normalisé créé par *MODNOM* et localisé dans MODCHP$(TAILLE).

Le reste de *MODNOM* fonctionne exactement de la même manière. On notera, toutefois, qu'en modifiant les autres zones on donne une valeur à RMOD sans ré-initialiser TRIE (voir ligne 14490, par exemple). C'est seulement en changeant la zone du nom que le fichier de données peut se trouver dérangé, puisqu'il est classé dans l'ordre des noms. En changeant toute autre zone, il sera simplement indiqué qu'un enregistrement a été modifié (RMOD=1) et que le fichier doit être sauvegardé lorsque le programme sera terminé.

L'autre routine exécutée est *EFFENR* — pour effacer un enregistrement. Elle est très simple. D'abord, elle efface l'écran (ligne 15020) et affiche un message expliquant ce qui se passe. Elle appelle ensuite *TROUVENR* pour localiser l'enregistrement qui doit être effacé. On a ensuite le choix entre appuyer sur RETURN pour effacer l'enregistrement ou sur la barre d'espacement pour revenir au menu principal. Un message d'avertissement est également affiché (ligne 15160). Une approche encore meilleure pourrait être de répondre par ÊTES-VOUS SÛR? si l'on a appuyé sur RETURN et de n'effacer l'enregistrement que si l'on tape la touche O pour « oui » (c'est-à-dire IF INKEY$="O" THEN...).

EFFENR ne ré-initialise pas le drapeau TRIE. Puisque le fichier est déjà dans l'ordre alphabétique, le fait d'effacer un enregistrement complet ne changera pas cet ordre. Par contre, le fichier a été modifié, et donc RMOD est ré-initialisé à la ligne 15340 et TAILLE est diminué d'une unité à la ligne 13550 pour tenir compte du fait que le fichier a maintenant un enregistrement en moins. Tous les enregistrements sont donc décalés de −1 aux lignes 15260 à 15320.

On peut aussi remarquer que *TROUVENR* inclut un appel conditionnel à un sous-programme appelé *LSTACT* pour imprimer l'enregistrement ACTuel localisé par *TROUVENR*. Si vous n'avez pas d'imprimante, remplacez simplement la ligne 13540 par un REM pour une exécution future et supprimez les lignes 13600 à 13690.

Variantes de basic

LPRINT

Cette commande n'est pas disponible sur le Commodore 64, le Vic-20, le BBC Micro ni le Dragon 32.

Sur le BBC Micro avec une imprimante parallèle insérer les lignes suivantes :

```
13605 VDU 2
13680 VDU 3
```

Elles mettent en service et hors service l'imprimante.
Substituer PRINT à LPRINT aux lignes 13610 à 13670. Pour plus d'informations, consulter le manuel de l'utilisateur.

Sur le Commodore, insérer ces lignes :

```
13605 OPEN 4,4:CMD 4
13680 PRINT # 4:CLOSE 4
```

Elles mettent en service et hors service l'imprimante.
Substituer PRINT à LPRINT aux lignes 13610 à 13670.

Sur le Dragon 32, insérer ces lignes :

```
13605 OPEN"O",-2
13680 CLOSE -2
```

Elles mettent en service et hors service l'imprimante.
Substituer PRINT -2, (la virgule fait partie de l'instruction) à LPRINT aux lignes 13610 à 13670.

SPECTRUM

Le programme de carnet d'adresses pour le Spectrum sera publié intégralement prochainement.

Programme de carnet d'adresses

```
10 REM « PROGRAMME PRINCIPAL »
20 REM "INITIL"
30 GOSUB 1000
40 REM "ACCUEIL"
50 GOSUB 3000
60 REM "CHOIX"
70 GOSUB 3500
80 REM "EXECUT"
90 GOSUB 4000
100 IF CHOI <> 9 THEN 60
110 END
1000 REM SOUS PROGRAMME "INITIL"
1010 GOSUB 1100: REM SOUS-PROGRAMME "CRETAB" (CRÉE DES TABLEAUX)
1020 GOSUB 1400: REM SOUS-PROGRAMME "LECFCR" (LIT LE FICHIER)
1030 GOSUB 1600: REM SOUS-PROGRAMME "DEFDRA" (DÉFINIT DES DRAPEAUX)
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM SOUS-PROGRAMME "CRETAB" (CRÉE DES TABLEAUX)
1110 DIM NOMCHP$(50)
1120 DIM MODCHP$(50)
1130 DIM RUECHP$(50)
1140 DIM VILCHP$(50)
1150 DIM CPOCHP$(50)
1160 DIM TELCHP$(50)
1170 DIM NDXCHP$(50)
1180 REM
1190 REM
1200 REM
1210 LET TAILLE = 0
1220 LET RMOD = 0
1230 LET TRIE = 1
1240 LET ACT = 0
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 RETURN
1400 REM SOUS-PROGRAMME "LECFCR"
1410 OPEN «I», #1, « ADBK.DAT »
1420 INPUT #1, TEST$
1430 IF TEST$ = «@PREMIER » THEN GOTO 1530: REM FERMER ET RETOURNER
1440 LET NOMCHP$(1) = TEST$
1450 INPUT #1; MODCHP$(1);RUECHP$(1);VILCHP$(1);CPOCHP$(1);TELCHP$(1)
1460 INPUT #1;NDXCHP$(1)
1470 LET TAILLE = 2
1480 FOR L = 2 TO 50
1490 INPUT #1; NOMCHP$(L);MODCHP$(L);RUECHP$(L);VILCHP$(L);CPOCHP$(L),
1500 INPUT #1; TELCHP$(L); NDXCHP$(L)
1510 LET TAILLE = TAILLE + 1
1520 IF EOF(1) = -1 THEN LET L = 50
1530 NEXT L
1540 CLOSE #1
1550 RETURN
1600 REM SOUS-PROGRAMME "DEFDRA"
1610 REM DÉFINIT DES DRAPEAUX APRÈS "LECFCR"
1620 REM
1630 REM
1640 IF TEST$ = «@PREMIER » THEN LET TAILLE = 1
1650 REM
1660 REM
1670 REM
1680 REM
1690 RETURN
3000 REM SOUS-PROGRAMME "ACCUEIL"
3010 PRINT CHR$(12):REM EFFACE ÉCRAN
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT
3060 PRINT TAB(12):« "VOICI LE PROGRAMME" »
3070 PRINT TAB(4): « "DE CARNET D'ADRESSES INFORMATISÉ" »
3080 PRINT TAB(10):« "DE ABC INFORMATIQUE" »
3090 PRINT
3100 PRINT « (APPUYER SUR BARRE D'ESPACEMENT POUR CONTINUER) »
3110 FOR L = 1 TO 1
3120 IF INKEY$ <>«» THEN L = 0
3130 NEXT L
3140 PRINT CHR$(12)
3150 RETURN
3500 REM SOUS-PROGRAMME "CHOIX"
3510 REM
3520 IF TEST$ = «@PREMIER » THEN GOSUB 380.REM SOUS-PROGRAMME "PREM"
3530 IF TEST$ = «@PREMIER » THEN RETURN
3540 REM "CHMENU"
3550 PRINT CHR$(12): REM EFFACE ÉCRAN
3560 PRINT « CHOISISSEZ UNE DES OPTIONS SUIVANTES »
3570 PRINT
3580 PRINT
3590 PRINT
3600 PRINT « 1: TROUVER UN ENREGISTREMENT (A PARTIR D'UN NOM) »
3610 PRINT « 2: TROUVER DES NOMS (A PARTIR D'UN NOM INCOMPLET) »
3620 PRINT « 3: TROUVER DES ENREGISTREMENTS (A PARTIR D'UNE VILLE) »
3630 PRINT « 4: TROUVER DES ENREGISTREMENTS (A PARTIR D'INITIALES) »
3640 PRINT « 5: LISTER TOUS LES ENREGISTREMENTS »
3650 PRINT « 6: AJOUTER UN NOUVEL ENREGISTREMENT »
3660 PRINT « 7: MODIFIER UN ENREGISTREMENT »
3670 PRINT « 8: EFFACER UN ENREGISTREMENT »
3680 PRINT « 9: QUITTER ET SAUVEGARDER »
3690 PRINT
3700 PRINT
3710 REM "ENTCHOI"
3720 REM
3730 LET L = 0
3740 LET I = 0
3750 FOR L = 1 TO 1
3760 PRINT « ENTRER UNE OPTION (1 A 9) »
3770 FOR I = 1 TO 1
3780 LET A$ = INKEY$
3790 IF A$ = «» THEN I = 0
3800 NEXT I
3810 LET CHOI = VAL(A$)
3820 IF CHOI <1 THEN L = 0
3830 IF CHOI >9 THEN L = 0
3840 NEXT L
3850 RETURN
3860 REM SOUS-PROGRAMME "PREM" (AFFICHE UN MESSAGE)
3870 LET CHOI = 6
3880 PRINT CHR$(12) : REM EFFACE ÉCRAN
3890 PRINT
3900 PRINT TAB (5) ; « IL N'Y A PAS D'ENREGISTREMENTS »
3910 PRINT TAB (6) ; « DANS LE FICHIER : VOUS DEVREZ »
3920 PRINT TAB (4) ; « D'ABORD AJOUTER UN ENREGISTREMENT »
3939 PRINT
3949 PRINT « (APPUYER SUR LA BARRE D'ESPACEMENT POUR CONTINUER) »
3950 FOR B = 1 TO 1
3960 IF INKEY$ <> «» THEN B = 0
3970 NEXT B
3980 PRINT CHR$ (12) : REM EFFACE ÉCRAN
3990 RETURN
4000 REM SOUS PROGRAMME "EXECUT"
4010 REM
4020 REM
4030 REM
4040 IF CHOI = 1 THEN GOSUB 13000 : REM "TROUVENR"
4050 REM 2 EST "TRNOMS"
4060 REM 3 EST "TRVILLE"
4070 REM 4 EST "TRINIT"
4080 REM 5 EST "LSTENR"
4090 IF CHOI = 6 THEN GOSUB 10000 : REM "AJOUTENR"
4100 IF CHOI = 7 THEN GOSUB 14000 : REM "MODENR"
4110 IF CHOI = 8 THEN GOSUB 15000 : REM "EFFENR"
4120 IF CHOI = 9 THEN GOSUB 11000 . REM "QUITTE"
4130 REM
4140 RETURN
10000 REM SOUS PROGRAMME "AJOUTENR"
10010 PRINT CHR$(12) : REM EFFACE ÉCRAN
10020 INPUT « ENTREZ LE NOM »;NOMCHP$ (TAILLE)
10030 INPUT « ENTREZ LA RUE »; RUECHP$ (TAILLE)
10040 INPUT « ENTREZ LE CODE POSTAL »; CPOCHP$ (TAILLE)
10050 INPUT « ENTREZ LA VILLE »; VILCHP$ (TAILLE)
10060 INPUT « ENTREZ LE NUMÉRO DE TÉLÉPHONE »; TELCHP$ (TAILLE)
10070 LET RMOD = 1 : LET TRIE = 1 : REM MODIFIE ET NON TRIE
10080 LET NDXCHP$ (TAILLE) = STR$ (TAILLE)
10090 LET TEST$ = « »
10100 GOSUB 10200 : REM "MODNOM"
10110 LET CHOI = 0
10120 LET TAILLE = TAILLE + 1
10130 REM
10140 REM
10150 RETURN
10200 REM SOUS-PROGRAMME "MODNOM"
10210 REM DÉPLACE LE CONTENU DE NOMCHP$ DANS LA CASE SUPÉRIEURE.
10220 REM ENLÈVE LE RESTE ET STOCKE DANS L'ORDRE .
10230 REM NOM + ESPACE + PRÉNOM DANS MODCHP$
10240 REM
10250 LET N$ = NOMCHP$ (TAILLE)
10260 FOR L = 1 TO LEN (N$)
10270 LET TEMP$ = MID$ (N$, L, 1)
10280 LET T = ASC (TEMP$)
10290 IF T> = 97 THEN T = T — 32
10300 LET TEMP$ = CHR$ (T)
10310 LET P$ = P$ + TEMP$
10320 NEXT L
10330 LET N$ = P$
10340 REM LOCALISE LE DERNIER ESPACE
10350 FOR L = 1 TO LEN (N$)
10360 IF MID$ (N$, L, 1) = « » THEN S = L
10370 NEXT L
10380 REM ENLÈVE LE RESTE ET STOCKE LE PRÉNOM
10390 REM EN CNOM$
10400 FOR L = 1 TO S — 1
10410 IF ASC (MID$ (N$, L, 1)) > 64 THEN CNOM$ = CNOM$ + MID$ (N$, L, 1)
10420 NEXT L
10430 REM ENLÈVE LE RESTE ET STOCKE LE NOM
10440 REM DANS SNOM$
10450 FOR L = S + 1 TO LEN (N$)
10460 IF ASC (MID$ (N$, L, 1)) > 64 THEN SNOM$ = SNOM$ + MID$ (N$, L, 1)
10470 NEXT L
10480 LET MODCHP$ (TAILLE) = SNOM$ + « » + CNOM$ « »
10490 LET P$ = « » : LET N$ = « » : LET SNOM$ = « » : LET CNOM$ = « » : LET S = 0 « »
10500 RETURN
11000 REM SOUS-PROGRAMME "QUITTE"
11010 REM TRIE ET SAUVEGARDE LE FICHIER
11020 REM SI UN ENREGISTREMENT A ÉTÉ
11030 REM MODIFIÉ (RMOD = 1)
11040 REM OU NON TRIE (TRIE = 0)
11050 REM RMOD = 0 ET TRIE = 0 EST ILLÉGAL
11060 REM
11070 IF RMOD = 0 AND TRIE = 1 THEN RETURN
11080 IF RMOD = 1 AND TRIE = 0 THEN GOSUB 11200 : REM "TRIENR"
11090 GOSUB 12000 : REM "SAUVENR"
11100 RETURN
11200 REM SOUS PROGRAMME "TRIENR"
11210 REM TRIE TOUS LES ENREGISTREMENTS SELON LEUR ZONE MODCHP$
11220 REM DANS L'ORDRE ALPHABÉTIQUE ET MET A JOUR LA ZONE NDXCHP$
11230 REM
11240 REM
11250 LET S = 0
11260 FOR L = 1 TO TAILLE — 2
11270 IF MODCHP$ (L) > MODCHP$ (L + 1) THEN GOSUB 11350
11280 NEXT L
11290 IF S = 1 THEN 11250
11300 REM
11310 REM
11320 LET TRIE = 1 : REM DRAPEAU 'FICHIER TRIE'
11330 REM
11340 RETURN
11350 REM SOUS-PROGRAMME "ECHENR"
11360 LET TNOMCH$ = NOMCHP$ (L)
11370 LET TMODCH$ = MODCHP$ (L)
11380 LET TRUECH$ = RUECHP$ (L)
11390 LET TVILCH$ = VILCHP$ (L)
```

```
11400 LET TCPOCH$ = CPOCHP$ (L)
11410 LET TTELCH$ = TELCHP$ (L)
11420 REM
11430 LET NOMCHP$ (L) = NOMCHP$ (L + 1)
11440 LET MODCHP$ (L) = MODCHP$ (L + 1)
11450 LET RUECHP$ (L) = RUECHP$ (L + 1)
11460 LET VILCHP$ (L) = VILCHP$ (L + 1)
11470 LET CPOCHP$ (L) = CPOCHP$ (L + 1)
11480 LET TELCHP$ (L) = TELCHP$ (L + 1)
11490 LET NDXCHP$ (L) = STR$ (L)
11500 REM
11510 LET NOMCHP$ (L + 1) = TNOMCHP$
11520 LET MODCHP$ (L + 1) = TMODCHP$
11530 LET RUECHP$ (L + 1) = TRUECHP$
11540 LET VILCHP$ (L + 1) = TVILCHP$
11550 LET CPOCHP$ (L + 1) = TCPOCHP$
11560 LET TELCHP$ (L + 1) = TTELCHP$
11570 LET NDXCHP$ (L + 1) = STR$ (L + 1)
11580 LET S = 1
11590 REM
11600 RETURN
12000 REM SOUS-PROGRAMME "SAUVENR"
12010 REM
12020 REM
12030 OPEN « O », #1, « ADBK.DAT »
12040 REM
12050 FOR L = 1 TO TAILLE — 1
12060 PRINT #1, NOMCHP$ (L); « , »; MODCHP$ (L); « , »; RUECHP$ (L); « , »;
12070 PRINT #1, CPOCHP$ (L); « , »; VILCHP$ (L); « , »; TELCHP$ (L); « , »;
      NDXCHP$ (L)
12080 NEXT L
12090 REM
12100 REM
12110 REM
12120 REM
12130 CLOSE #1
12140 REM
12150 RETURN
13000 REM SOUS-PROGRAMME "TROUVENR" (TROUVE UN ENREGISTREMENT)
13010 PRINT CHR$ (12) : REM EFFACE ÉCRAN
13020 REM
13030 IF TRIE = 0 THEN GOSUB 11200 : REM "TRIENR"
13040 PRINT
13050 PRINT
13060 PRINT TAB (6); « RECHERCHE D'UN ENREGISTREMENT »
13070 PRINT TAB (10); « A PARTIR DU NOM »
13080 PRINT
13090 PRINT TAB (6); « TAPER LE NOM COMPLET »
13100 PRINT TAB (5); « EN COMMENÇANT PAR LE PRÉNOM »
13110 PRINT
13120 PRINT
13130 REM
13140 INPUT « LE NOM EST »; NOMCHP$ ( TAILLE)
13150 GOSUB 10200 : REM SOUS-PROGRAMME "MODNOM"
13160 LET CLERECH$ = MODCHP$ (TAILLE)
13170 REM
13180 REM
13190 REM
13200 REM
13210 REM
13220 LET INF = 1
13230 LET SUP = TAILLE — 1
13240 FOR L = 1 TO 1
13250 LET MED = INT ((INF + SUP)/2)
13260 IF MODCHP$ (MED) <> CLERECH$ THEN L = 0
13270 IF MODCHP$ (MED) < CLERECH$ THEN INF = MED + 1
13280 IF MODCHP$ (MED) > CLERECH$ THEN SUP = MED - 1
13290 IF INF > SUP THEN L = 1
13300 NEXT L
13310 REM
13320 IF INF > SUP THEN LET ACT = 0
13330 IF INF < = SUP THEN LET ACT = MED
13340 IF ACT = 0 THEN GOSUB 13700 : REM "PASENR"
13350 IF ACT = 0 THEN RETURN
13360 REM
13370 REM
13380 PRINT CHR$ (12) : REM EFFACE ÉCRAN
13390 PRINT
13400 PRINT TAB (10); « "ENREGISTREMENT TROUVÉ" »
13410 PRINT
13420 PRINT « NOM : »; NOMCHP$ (ACT)
13430 PRINT « RUE : »; RUECHP$ (ACT)
13440 PRINT « CODE POSTAL : »; CPOCHP$ (ACT)
13450 PRINT « VILLE : »; VILCHP$ (ACT)
13460 PRINT « TEL : »; TELCHP$ (ACT)
13470 PRINT
13480 PRINT TAB (4); « TAPER UNE LETTRE POUR IMPRIMER »
13490 PRINT TAB (4); « OU LA BARRE D'ESPACEMENT POUR CONTINUER »
13500 FOR I = 1 TO 1
13510 LET A$ = INKEY$
13520 IF A$ = « » THEN I = 0
13530 NEXT I
13540 IF A$ <> « » THEN GOSUB 13600 : REM "LSTACT"
13550 RETURN
13600 REM SOUS-PROGRAMME "LSTACT" (LISTE L'ENREGISTREMENT ACTUEL)
13610 LPRINT
13620 LPRINT « NOM : »; NOMCHP$ (ACT)
13630 LPRINT « RUE : »; RUECHP$ (ACT)
13640 LPRINT « CODE POSTAL : »; CPOCHP$ (ACT)
13650 LPRINT « VILLE : »; VILCHP$ (ACT)
13660 LPRINT « TEL : »; TELCHP$ (ACT)
13670 LPRINT
13680 LPRINT
13690 RETURN
13700 REM SOUS-PROGRAMME "PASENR" (ENREGISTREMENT NON TROUVÉ)
13710 PRINT CHR$ (12) : REM EFFACE ÉCRAN
13720 PRINT TAB (8); « "ENREGISTREMENT NON TROUVÉ" »
13730 PRINT TAB (3); « "SOUS LA FORME : »; NOMCHP$ (TAILLE); «"»
13740 PRINT
13750 PRINT « (APPUYER SUR LA BARRE D'ESPACEMENT POUR CONTINUER) »
13760 FOR I = 1 TO 1
13770 IF INKEY$ <> « » THEN I = 0
13780 NEXT I
13790 RETURN
14000 REM SOUS-PROGRAMME "MODENR" (MODIFIE UN ENREGISTREMENT)
14010 REM
14020 PRINT CHR$ (12) : REM EFFACE ÉCRAN
14030 PRINT
14040 PRINT
14050 PRINT
14060 PRINT
14070 PRINT TAB (7); « "POUR MODIFIER UN ENREGISTREMENT" »
14080 PRINT TAB (10); « "LOCALISEZ D'ABORD" »
14090 PRINT TAB (8); « "L'ENREGISTREMENT SOUHAITÉ" »
14100 REM
14110 REM
14120 GOSUB 13030 : REM SOUS-PROGRAMME "TROUVENR" SANS EFFACER ÉCRAN
14130 IF ACT = 0 THEN RETURN : REM ENREGISTREMENT NON TROUVÉ
14140 PRINT
14150 PRINT TAB (5); « VOULEZ-VOUS MODIFIER LE NOM ? »
14160 PRINT
14170 PRINT TAB (3); « TAPER RETURN POUR ENTRER UN NOUVEAU NOM »
14180 PRINT TAB (3); « OU LA BARRE D'ESPACEMENT POUR ZONE SUIVANTE »
14190 FOR I = 1 TO 1
14200 LET A$ = INKEY$
14210 IF A$ <> CHR$ (13) AND A$ <> « » THEN I = 0
14220 NEXT I
14230 IF A$ = CHR$ (13) THEN INPUT « NOUVEAU NOM »; NOMCHP$ (ACT)
14240 IF A$ = CHR$ (13) THEN RMOD = 1
14250 IF A$ = CHR$ (13) THEN TRIE = 0
14260 IF A$ = CHR$ (13) THEN NOMCHP$ (TAILLE) = NOMCHP$ (ACT)
14270 IF A$ = CHR$ (13) THEN GOSUB 10200 : REM " MODNOM"
14280 IF A$ = CHR$ (13) THEN LET MODCHP$ (ACT) = MODCHP$ (TAILLE)
14290 PRINT
14300 PRINT TAB (5); « VOULEZ-VOUS MODIFIER LA RUE ? »
14310 PRINT
14320 PRINT « TAPER SUR RETURN POUR CHANGER LA RUE »
14330 PRINT « OU LA BARRE D'ESPACEMENT POUR LA ZONE SUIVANTE »
14340 FOR I = 1 TO 1
14350 LET A$ = INKEY$
14360 IF A$ <> CHR$ (13) AND A$ <> « » THEN I = 0
14370 NEXT I
14380 IF A$ = CHR$ (13) THEN RMOD = 1
14390 IF A$ = CHR$ (13) THEN INPUT « NOUVELLE RUE »; RUECHP$ (ACT)
14400 PRINT
14410 PRINT TAB (7); « VOULEZ-VOUS MODIFIER LA VILLE ? »
14420 PRINT
14430 PRINT « TAPER SUR RETURN POUR CHANGER LA VILLE »
14440 PRINT « OU LA BARRE D'ESPACEMENT POUR LA ZONE SUIVANTE »
14450 FOR I = 1 TO 1
14460 LET A$ = INKEY$
14470 IF A$ <> CHR$ (13) AND A$ <> « » THEN I = 0
14480 NEXT I
14490 IF A$ = CHR$ (13) THEN RMOD = 1
14500 IF A$ = CHR$ (13) THEN INPUT « NOUVELLE VILLE »; VILCHP$ (ACT)
14510 PRINT
14520 PRINT TAB (4); « VOULEZ-VOUS MODIFIER LE CODE POSTAL ? »
14530 PRINT
14540 PRINT « TAPER SUR RETURN POUR CHANGER LE CODE POSTAL »
14550 PRINT « OU SUR LA BARRE D'ESPACEMENT POUR LA ZONE SUIVANTE »
14560 FOR I = 1 TO 1
14570 LET A$ = INKEY$
14580 IF A$ <> CHR$ (13) AND A$ <> « » THEN I = 0
14590 NEXT I
14600 IF A$ = CHR$ (13) THEN RMOD = 1
14610 IF A$ = CHR$ (13) THEN INPUT « NOUVEAU CODE POSTAL »; CPOCHP$ (ACT)
14620 PRINT
14630 PRINT « VOULEZ-VOUS MODIFIER LE NUMÉRO DE TÉLÉPHONE ? »
14640 PRINT
14650 PRINT « TAPER SUR RETURN POUR CHANGER LE NO DE TÉL. »
14660 PRINT « OU SUR LA BARRE D'ESPACEMENT POUR CONTINUER »
14670 FOR I = 1 TO 1
14680 LET A$ = INKEY$
14690 IF A$ <> CHR$ (13) AND A <> « » THEN I = 0
14700 NEXT I
14710 IF A$ = CHR$ (13) THEN RMOD = 1
14720 IF A$ = CHR$ (13) THEN INPUT « NOUVEAU NUMÉRO »; TELCHP$ (ACT)
14730 REM
14740 REM
14750 RETURN
15000 REM SOUS-PROGRAMME "EFFENR" (EFFACE UN ENREGISTREMENT)
15010 REM
15020 PRINT CHR$ (12) : REM EFFACE ÉCRAN
15030 PRINT
15040 PRINT
15050 PRINT
15060 PRINT
15070 PRINT TAB (6); « "POUR EFFACER UN ENREGISTREMENT" »
15080 PRINT TAB (2); « "LOCALISEZ D'ABORD L'ENREGISTREMENT SOUHAITÉ" »
15090 REM
15100 REM
15110 REM
15120 GOSUB 13030 : REM SOUS-PROGRAMME "TROUVENR" SANS EFFACER ÉCRAN
15130 IF ACT = 0 THEN RETURN : REM ENREGISTREMENT NON TROUVÉ
15140 PRINT
15150 PRINT « VOULEZ-VOUS EFFACER CET ENREGISTREMENT ? »
15160 PRINT TAB (4); « "ATTENTION"... C'EST VOTRE DERNIÈRE CHANCE »
15170 PRINT
15180 PRINT TAB (6); « TAPEZ SUR RETURN POUR EFFACER »
15190 PRINT TAB (4); « OU SUR LA BARRE D'ESPACEMENT POUR CONTINUER »
15200 FOR I = 1 TO 1
15210 LET A$ = INKEY$
15220 IF A$ <>CHR$(13) AND A$ <> « » THEN I = 0
15230 NEXT I
15240 IF A$ = « » THEN RETURN
15250 FOR L = ACT TO TAILLE — 2
15260 LET NOMCHP$ (L) = NOMCHP$ (L + 1)
15270 LET MODCHP$ (L) = MODCHP$ (L + 1)
15280 LET RUECHP$ (L) = RUECHP$ (L + 1)
15290 LET VILCHP$ (L) = VILCHP$ (L + 1)
15300 LET CPOCHP$ (L) = CPOCHP$ (L + 1)
15310 LET TELCHP$ (L) = TELCHP$ (L + 1)
15320 LET NDXCHP$ (L) = STR$ (L)
15330 NEXT L
15340 LET RMOD = 1
15350 LET TAILLE = TAILLE - 1
15360 REM
15370 REM
15380 REM
15390 RETURN
```

Grace Hopper

Grace Hopper fut l'une des premières femmes à travailler sur les ordinateurs, et contribua à la création des langages évolués.

Sperry

On considère souvent l'informatique comme un domaine strictement masculin. Pourtant les femmes y jouent un rôle sans cesse croissant, et de ce point de vue Grace Hopper fut l'une des pionnières. Sa contribution la plus importante se situe au niveau de la programmation : elle créa le premier compilateur et participa à la mise au point du langage COBOL.

Une fois ses diplômes obtenus, Grace Hopper travailla d'abord à Yale, puis revint à Vassar, son université d'origine, pour y enseigner les mathématiques. Elle y demeura jusqu'à l'âge de trente-neuf ans ; il lui fut alors demandé de mettre ses compétences au service de l'effort de guerre, et elle travailla à divers projets de calcul pour le compte de la marine. En 1945, elle fut envoyée à Harvard pour collaborer avec le physicien Howard Aiken à la mise au point d'un ordinateur. Aiken avait contacté IBM en ce sens dès 1937 ; son projet était de construire un appareil qui aurait recours au matériel mécanographique classique. L'engin était encore de conception strictement mécanique, mais il était assez prometteur pour qu'IBM accepte d'investir dans la fabrication d'un modèle amélioré qui utiliserait des relais électromécaniques. Ce devait être le Harvard Mark II.

En ces temps héroïques, le seul moyen de programmer l'ordinateur était de le recâbler à chaque nouvelle opération. Au cours de l'été torride de 1945, Grace Hopper se retrouva bientôt submergée sous les câbles. Les militaires avaient impérativement besoin de nombreux calculs balistiques, et périodiquement le commandant Aiken passait en demandant : « Hopper, et mes nombres? » Après une panne particulièrement épineuse, dont la cause était une phalène entrée par la fenêtre et qui avait trouvé sa fin dans un relais, Grace répondit simplement : « Nous faisons la chasse aux insectes ! » (“*We are debugging!*”). Là est l'origine du mot *bug*, qui désigne en anglais une erreur de programmation (en français : « bogue ») et dont le sens premier est insecte. Le journal de bord du Harvard Mark II, conservé au Musée naval de Virginie, contient encore, à la date du 9 septembre 1945, 14 h 45, la toute première « bug », collée dans la marge après avoir été soigneusement récupérée avec une pince à épiler.

Deux ingénieurs, John Mauchly et Presper Eckert, construisaient à la même époque un autre ordinateur, l'ENIAC. Après la guerre ils montèrent leur propre affaire, afin d'assurer la production d'une version de cette machine tournée vers le monde des affaires. C'est avec ces intentions qu'ils demandèrent à Grace de se joindre à eux. L'appareil s'appelait l'UNIVAC (UNIVersal ACcounting machine, « machine à calculer universelle »). Grace s'occupa surtout de la programmation, et c'est en cherchant à mettre au point des programmes de gestion qu'elle eut l'idée de simplifier la réécriture de certaines routines de fonctionnement qui revenaient sans cesse. Faire écrire à l'ordinateur ses *propres* programmes était, pour l'époque, une trouvaille remarquable : Grace créa ainsi le premier langage de programmation et le compilateur qui devait en assurer la traduction en code machine. Il fut simplement appelé « A-O », et provoqua l'incrédulité des spécialistes, pour qui les ordinateurs pouvaient tout au plus faire des calculs arithmétiques ou manipuler certains symboles. Ils furent stupéfaits de voir l'appareil passer de lui-même à un sous-programme conservé en mémoire.

En mai 1959, le capitaine Hopper fut invitée par le Pentagone à se joindre à un comité de travail qui devait créer et standardiser un langage unique destiné aux ordinateurs de gestion. En moins d'un an cet organisme mit au point la première version du COBOL (COmmon Business Oriented Langage). Grace avait joué un rôle important dans cette tentative de retenir les points forts de chaque langage existant, et d'offrir ainsi au monde des affaires un outil particulièrement performant. Le COBOL est aujourd'hui encore très largement utilisé dans le monde des affaires — signe évident de la réussite de l'entreprise.

COBOL
Le COBOL fut l'un des premiers langages explicitement prévus à l'intention des non-mathématiciens. Il favorise l'emploi de procédures générales rédigées dans un style de type narratif, plutôt que de recourir à des routines codées applicables à un seul type de problème.
Un programme écrit en COBOL comporte quatre divisions. Identification : nom du programme, nom de l'auteur, références générales. Environnement : tous les détails relatifs au type d'ordinateur utilisé (bien que le COBOL permette de rédiger des programmes « translatables », tournant sur des appareils très différents). Données : employées dans de nombreuses parties du même programme. Procédures : ensemble des opérations de traitement des données.

PROGRAMME N° 10

CENTRONS LE CROCODILE

Reprenons le programme de la semaine précédente, mais imposons quelques contraintes aux coordonnées X et Y.
Nous allons maintenant saisir les coordonnées X et Y grâce à un INPUT. Pour éviter que le crocodile soit à l'extérieur de l'écran, introduisons des contrôles sur les valeurs de X et Y.
Cela donne :

```
 5 REM DESSINONS UN CROCODILE CENTRÉ
10 PRINT « COORDONNÉES DU CROCO »
11 PRINT
12 INPUT « X = »; X
13 PRINT : INPUT « Y = »; Y
14 REM CONTROLE DES COORDONNÉES DU CROCO
15 IF X < 2 THEN X = 2
20 IF X > 37 THEN X = 37
25 IF Y < 1 THEN Y = 1
30 IF Y > 38 THEN Y = 38
40 REM DESSIN DU CROCO
45 GR
50 COLOR = 7
60 PLOT X - 1, Y - 1
70 HLIN X - 1, X + 2 AT Y
80 COLOR = 9
90 PLOT X - 1, Y + 1
100 PLOT X + 1, Y + 1
110 COLOR = 15
120 PLOT X - 2, Y - 1
```

Question :

Demandez-vous pourquoi la valeur maximale de X est 37 alors que la valeur maximale de Y est 38.
Dans ce cas, si vous demandez à la machine de tracer un crocodile hors des limites de l'écran, l'ordinateur le dessinera automatiquement sur le bord le plus proche. Vous pourriez aussi lui demander d'indiquer un message d'erreur. La solution précédente présente l'avantage de ne pas arrêter le programme.
Il est aussi possible d'affecter à X et Y des valeurs aléatoires grâce aux deux lignes d'instruction

```
X = INT (RND(1)*37 + 1
Y = INT (RND(1)*37) + 1
```

```
 5 DESSINONS UN CROCODILE CENTRÉ
11 X = INT (RND(1)*37) + 1
12 Y = INT (RND(1)*37) + 1
14 REM CONTROLE DES COORDONNÉES DU CROCO
15 IF X < 2 THEN X = 2
20 IF X > 37 THEN X = 37
25 IF Y < 1 THEN Y = 1
30 IF Y > 38 THEN Y = 38
40 REM DESSIN DU CROCO
45 GR
50 COLOR = 7
60 PLOT X - 1, Y - 1
70 HLIN X - 1, X + 2 AT Y
80 COLOR = 9
90 PLOT X - 1, Y + 1
100 PLOT X + 1, Y + 1
110 COLOR = 15
120 PLOT X - 2, Y - 1
```

L'organigramme est le même que le précédent.

Rappel :
Nous avons vu précédemment l'instruction COLOR = N° de couleur pour déterminer le choix de la couleur d'un point.
16 couleurs sont disponibles au total en mode GR :

0	Noir
1	Magenta
2	Bleu foncé
3	Pourpre
4	Vert foncé
5	Gris
6	Bleu ciel
7	Bleu clair
8	Marron
9	Orangé
10	Gris
11	Rose
12	Vert clair
13	Jaune
14	Turquoise
15	Blanc

Voici un petit programme pour obtenir un échantillon de chaque couleur disponible :

```
$LIST

 5 REM PASSAGE EN MODE GR
10 GR
15 REM BOUCLE SUR LES NUMEROS DE COULEURS
20 FOR I = 0 TO 15
30 COLOR = I
35 REM AFFICHAGE D'UN POINT DE COULEUR I
40 PLOT I + 2, 10
60 INPUT "TAPER RETURN POUR UNE AUTRE
   COULEUR";Z$
62 REM COULEUR SUIVANTE
65 NEXT I
68 REM ON REPASSE EN MODE TEXT
70 TEXT
```

Sachons donc que pour « allumer » un point A de coordonnées (X,Y) en couleur bleu on tape :

```
COLOR = 2
PLOT X,Y
```

Pour éteindre le même point on écrit :

```
COLOR = 0
PLOT X,Y
```

Il n'existe pas d'instruction spécifique pour l'extinction d'un point. Pour « éteindre » un point, il faut lui redonner la couleur du fond (noir).

Créons le mouvement

En éteignant ou en rallumant ce point on peut aussi créer le clignotement.
En éteignant un point après l'avoir allumé, puis en allumant un point immédiatement voisin, on créé l'effet de déplacement.

Voici un petit programme créant le déplacement.

```
$LIST

 4  C = 1
 5  REM PASSAGE EN MODE GR
10  GR
30  FOR I = 1 TO 39
40  COLOR = C
42  REM DESSIN EN COULEUR C DU POINT
45  PLOT I, I
48  REM BOUCLE DE TEMPORISATION
50  FOR ST = 1 TO 40:NEXT ST
55  REM EXTINCTION DU POINT
60  COLOR = 0:PLOT I, I
70  NEXT I
80  INPUT "TAPER RETURN POUR
    CONTINUER", Z$
82  REM CHANGEMENT DE COULEUR
85  C = C + 1
87  REM ON RECOMMENCE APRES CHANGEMENT
    DE COULEUR
90  GOTO 30
95  REM RETOUR AU MODE TEXT
100 TEXT
```

Une dernière instruction en mode GR :

SCRN (comme *screen*, « écran »).

SCRN teste la couleur d'un point. Cette instruction permet surtout de déterminer si un point est « allumé » ou de la couleur du fond.

Un petit exemple :

```
$LIST

 5 REM PASSAGE EN MODE TEXT
10 GR
15 REM CHOIX DE COULEUR
20 COLOR = 2
25 REM ON DESSINE LE POINT
30 PLOT 30, 30
35 REM LA MACHINE ECRIT LE NUMERO
   DE LA COULEUR DU POINT DESSINE
40 PRINT SCRN (30, 30)
50 INPUT "TAPER RETURN POUR RETOUR
   AU MODE TEXT";Z$
60 TEXT


$RUN
2

TAPER RETURN POUR RETOUR AU MODE TEXT
```